

## Secure management of keys using control vectors.

**Publication number:** DE68926005T

**Publication date:** 1996-10-17

**Inventor:** MATYAS STEPHEN M (US); ABRAHAM DENNIS G (US); JOHNSON DONALD B (US); KARNE RAMESH K (US); LE AN V (US); PRYMAK ROSTISLAW (US); THOMAS JULIAN (US); WILKINS JOHN D (US); YEH PHIL C (US); SMITH RONALD M (US); WHITE STEVE R (US); ARNOLD WILLIAM C (US)

**Applicant:** IBM (US)

**Classification:**

- International: G07F7/10; H04L9/00; H04L9/08; H04L9/32; G07F7/10; H04L9/00; H04L9/08; H04L9/32; (IPC1-7): H04L9/08

- European: G07F7/10E; H04L9/00; H04L9/08C; H04L9/32B

**Application number:** DE19896026005T 19890809

**Priority number(s):** US19880231114 19880811; US19880238010 19880829; US19880237938 19880826; US19880233515 19880818

**Also published as:**

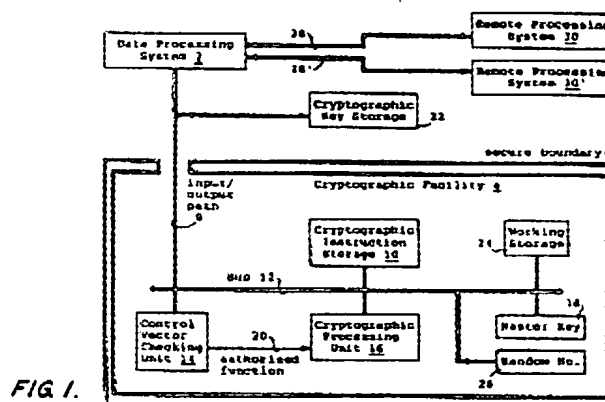
EP0356065 (A)  
EP0356065 (A)  
EP0356065 (B)

Report a data error he

Abstract not available for DE68926005T

Abstract of corresponding document: **EP0356065**

Arrangements are disclosed for validating that key management functions requested for a cryptographic key by the program have been authorised by the originator of the key. The invention includes a cryptographic facility characterised by a secure boundary through which passes an input path for receiving the cryptographic service requests, cryptographic keys and their associated control vectors, and an output path for providing responses thereto. There can be included within the boundary a cryptographic instruction storage coupled to the input path, a control vector checking unit and a cryptographic processing unit coupled to the instruction storage, and a master key storage coupled to the processing means, for providing a secure location for executing key management functions in response to the received service requests. The cryptographic instruction storage receives over the input path a cryptographic service request for performing a key management function on a cryptographic key. The control vector checking unit has an input coupled to the input path for receiving a control vector associated with the cryptographic key and an input connected to the cryptographic instruction storage, for receiving control signals to initiate checking that the control vector authorises the key management function which is requested by the cryptographic service request. The control vector checking unit has an authorisation output connected to an input of the cryptographic processing means, for signalling that the key management function is authorised, the receipt of which by the cryptographic processing unit



initiates the performance of the requested key management function with the cryptographic key. The invention enables the flexible control of many cryptographic key management functions in the generation, distribution and use of cryptographic keys, while maintaining a high security standard.

---

Data supplied from the **esp@cenet** database - Worldwide



①⑨ BUNDESREPUBLIK  
DEUTSCHLAND



DEUTSCHES  
PATENTAMT

⑫ Übersetzung der  
europäischen Patentschrift

⑧⑦ EP 0 356 065 B1

⑩ DE 689 26 005 T 2

⑤① Int. Cl.<sup>8</sup>:  
H 04 L 9/08

②① Deutsches Aktenzeichen: 689 26 005.9  
⑧⑥ Europäisches Aktenzeichen: 89 308 068.9  
⑧⑥ Europäischer Anmeldetag: 9. 8. 89  
⑧⑦ Erstveröffentlichung durch das EPA: 28. 2. 90  
⑧⑦ Veröffentlichungstag  
der Patenterteilung beim EPA: 20. 3. 96  
④⑦ Veröffentlichungstag im Patentblatt: 17. 10. 96

DE 689 26 005 T 2

③① Unionspriorität: ③② ③③ ③①

11.08.88 US 231114 29.08.88 US 238010  
26.08.88 US 237938 18.08.88 US 233515

⑦③ Patentinhaber:

International Business Machines Corp., Armonk,  
N.Y., US

⑦④ Vertreter:

Teufel, F., Dipl.-Phys., Pat.-Anw., 70569 Stuttgart

⑧④ Benannte Vertragsstaaten:

DE, FR, GB, IT, NL

⑦② Erfinder:

Matyas, Stephen M., Manassas, VA 22110, US;  
Abraham, Dennis G., Concord North Carolina, US;  
Johnson, Donald B., Manassas, VA 22111, US;  
Karne, Ramesh K., Herndon, VA 22071, US; Le, An  
V., Arlington, VA 22204, US; Prymak, Rostislaw,  
Dumfries Virginia, US; Thomas, Julian,  
Poughkeepsie, NY 12603, US; Wilkins, John D.,  
Somerville, VA 22739, US; Yeh, Phil C.,  
Poughkeepsie, NY 12603, US; Smith, Ronald M.,  
Wappingers Falls, NY 12590, US; White, Steve R.,  
New York, NY, US; Arnold, William C., Mahopac, NY  
10541, US

⑤④ Sichere Schlüsselverwaltung mittels Steuervektoren

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patentamt inhaltlich nicht geprüft.

DE 689 26 005 T 2

## B E S C H R E I B U N G

## SICHERE SCHLÜSSELVERWALTUNG MITTELS STEUERVEKTOREN

Die vorliegende Erfindung betrifft die sichere Schlüsselverwaltung mittels Steuervektoren.

Die kryptographische Umwandlung von Daten ist gewöhnlich durch einen ausgewählten Algorithmus oder eine Prozedur definiert. Da der Algorithmus in der Regel allgemein bekannt ist, hängt die Sicherheit der umgewandelten, d.h. verschlüsselten, Daten von der Geheimhaltung des Schlüssels ab. Der Schlüssel muß also geheimgehalten werden, damit ein Gegner nicht einfach den bekannten Algorithmus und den Schlüssel zur Dechiffrierung der Daten verwenden kann. Die Sicherheit der Daten steht und fällt deshalb mit der Sicherheit von sicheren Schlüsseln.

Die Schlüsselverwaltung umfaßt die Vorrichtungen, Funktionen und Prozeduren in einem Chiffriersystem, die dazu dienen, Chiffrierschlüssel und schlüsselbezogene Informationen so zu verwalten, daß die Sicherheit und Integrität der Schlüssel gewährleistet ist.

Zur Unterstützung der primären Verschlüsselungsanforderungen eines Benutzers oder eines Hostsystems verfügt die Schlüsselverwaltungsvorrichtung des Systems in der Regel über verschiedene Fähigkeiten einschließlich Installation, Speicherung und Generierung der Schlüssel sowie Schlüsselverteilung für den Import und den Export von Schlüsseln.

In allen Fällen ist das allgemeine Ziel die Verhinderung unbefugter Ermittlung oder Veränderung von Chiffrierschlüsseln.

Da chiffrierte Daten zwischen Systemen mit dem gleichen Chiffrieralgorithmus ausgetauscht werden können, kann die Möglichkeit der Versendung eines oder mehrerer geheimer Schlüssel



erforderlich sein. Zum Schutz der Vertraulichkeit eines geheimen Schlüssels während der Übermittlung vom Urheber zum vorgesehenen Empfänger muß der Schlüssel selber mit einem anderen geheimen Schlüssel (der dem Sender und dem Empfänger bereits bekannt ist) verschlüsselt werden. Es müssen Schlüsselverteilungsprotokolle definiert werden, um den kompatiblen sicheren Austausch von Schlüsseln zwischen Chiffriersystemen zu ermöglichen.

Die Speicherung von Schlüsseln auf einem unsicheren Medium (d.h. die Speicherung außerhalb eines sicheren Bereichs) macht es erforderlich, die Schlüssel selber zu chiffrieren. Beim Hauptschlüsselprinzip werden alle von einem Chiffriersystem verwendeten Schlüssel in verschlüsselter Form unter einem einzigen Schlüssel, dem sogenannten Hauptschlüssel, gespeichert. Der Hauptschlüssel selbst muß vor unbefugter Dechiffrierung und vor Veränderung geschützt werden. In der Regel stehen nichtkryptographische Mittel zum Schutz des Hauptschlüssels zur Verfügung (z.B. eine physische Zugriffskontrolle).

Derzeitige Verfahren zur Generierung und Verteilung von Chiffrierschlüsseln werden in dem Artikel im ICL Technical Journal, Bd. 4, Nr. 2, November 1984, S. 146 - 158, beschrieben.

¶ Nach dem Stand der Technik gibt es kein praktisches und flexibles Schlüsselverwaltungssystem mit hohen Datensicherheitsstandards.

Ein Ziel der vorliegenden Erfindung ist deshalb die Integrität des Chiffrieralgorithmus und bestimmter höherer Chiffrierfunktionen. Dieses Ziel erreicht die Erfindung gemäß Anspruch 1.

Mittels der vorliegenden Erfindung kann die Integrität gespeicherter oder verteilter Schlüssel gesichert, der Mißbrauch gespeicherter oder verteilter Schlüssel (wie z.B. die Verwendung eines Vertraulichkeitsschlüssels als Identifikationsüberprüfungsschlüssel) verhindert, die Verwendung gespeicherter oder

verteilter Schlüssel eingeschränkt (z.B. nur zur Identifikationsüberprüfung), eine sichere Installation des Hauptschlüssels und anderer manuell installierter Schlüsselchiffrierschlüssel gewährleistet und die Generierung neuer Schlüssel- und Datenchiffrierschlüssel abgesichert werden.

Im folgenden wird ein Ausführungsbeispiel der vorliegenden Erfindung beschrieben, das hier als Chiffrierarchitektur (CA) bezeichnet wird und als in einem Datenverarbeitungssystem implementiert erläutert wird. Das System führt ein Programm aus, das Chiffrierserviceanforderungen zur Verwaltung von Chiffrierschlüsseln ausgibt, die Steuervektoren zugeordnet sind, welche die Funktionen festlegen, zu welchen Funktionen die einzelnen Schlüssel von ihrem Urheber ermächtigt wurden. Bei diesem Verfahren wird geprüft, ob die Schlüsselverwaltungsfunktionen, die vom Programm für einen Chiffrierschlüssel angefordert wurden, vom Urheber des Schlüssels autorisiert sind.

Die Chiffriervorrichtung ist durch eine sichere Grenze gekennzeichnet, durch die ein Eingangspfad zum Empfang der Chiffrierserviceanforderungen der Chiffrierschlüssel und der zugeordneten Steuervektoren sowie ein Ausgangspfad für die entsprechenden Antworten verläuft. Innerhalb der Grenze können sich ein an den Eingangspfad gekoppelter Chiffrieranweisungsspeicher, ein Mittel zur Prüfung der Steuervektoren, ein an den Anweisungsspeicher gekoppeltes Chiffrierverarbeitungsmittel sowie ein an das Verarbeitungsmittel gekoppelter Hauptschlüsselspeicher, in dem Schlüsselverwaltungsfunktionen als Antwort auf die empfangenen Serviceanforderungen sicher ausgeführt werden können, befinden.

Der Chiffrieranweisungsspeicher empfängt über den Eingangspfad eine Chiffrierserviceanforderung zur Ausführung einer Schlüsselverwaltungsfunktion mit einem Chiffrierschlüssel. Das Mittel zur Prüfung des Steuervektors hat einen an den Eingangspfad angeschlossenen Eingang zum Empfang eines dem Chiffrierschlüssel zugeordneten Steuervektors und einen an den Chiffrieranweisungs-

speicher angeschlossenen Eingang zum Empfang der Steuersignale zum Starten der Prüfung, ob der Steuervektor zu der Schlüsselverwaltungsfunktion berechtigt, die von der Chiffrierserviceanforderung angefordert wurde.

Das System enthält ferner ein Mittel zur Speicherung von Chiffrierschlüsseln, das über den Eingangs- und den Ausgangspfad an die Verschlüsselungsvorrichtung angeschlossen ist. In diesem Speichermittel wird der Chiffrierschlüssel in einer verschlüsselten Form gespeichert, bei der der Chiffrierschlüssel unter dem Speicherschlüssel chiffriert ist, der ein logisches Produkt des zugeordneten Steuervektors und eines im Hauptschlüsselspeicher gespeicherten Hauptschlüssels ist.

Der Chiffrieranweisungsspeicher kann beispielsweise über den Eingangspfad eine Chiffrierserviceanforderung zur Wiederherstellung des Chiffrierschlüssels aus dem Chiffrierschlüsselspeichermittel empfangen, und das Mittel zur Überprüfung des Steuervektors sendet dann als Antwort darauf ein Berechtigungssignal an das Chiffrierverarbeitungsmittel, das besagt, daß die Funktion zur Wiederherstellung des Chiffrierschlüssels zugelassen ist. Das Chiffrierverarbeitungsmittel empfängt dann als Antwort auf das Berechtigungssignal die verschlüsselte Form des Chiffrierschlüssels aus dem Chiffrierschlüsselspeichermittel und dechiffriert die verschlüsselte Form unter dem Speicherschlüssel, der ein logisches Produkt aus dem zugeordneten Steuervektor und dem im Hauptschlüsselspeicher gespeicherten Hauptschlüssel ist.

Der Steuervektor enthält Felder, in denen zulässige Arten von Chiffrierfunktionen einschließlich Schlüsselverwaltungsfunktionen, Datenchiffrier- und Datendechiffrierfunktionen und Funktionen zur Verarbeitung persönlicher Identifikationszahlen festgelegt sind. Der zugehörige Steuervektor enthält außerdem Felder, in denen die Exportkontrolle und die Verwendung der Schlüssel definiert ist. Diese verschiedenen Felder erzwingen die Trennung verschiedener Schlüsselarten mit Verwendungszwecken, die sich

gegenseitig ausschließen sollten, damit die Sicherheit des Systems gewährleistet ist.

Das beschriebene Verfahren ermöglicht die flexible Steuerung zahlreicher Chiffrierschlüssel-Verwaltungsfunktionen bei der Generierung, Verteilung und Verwendung von Chiffrierschlüsseln unter Wahrung eines hohen Sicherheitsstandards.

Unter einem anderen Gesichtspunkt liefert die vorliegende Erfindung offensichtlich auch ein System oder eine Vorrichtung zur Durchführung des Verfahrens.

Im folgenden wird die vorliegende Erfindung mit Hilfe der beigefügten Zeichnungen anhand eines Ausführungsbeispiels exemplarisch beschrieben.

Fig. 1 ist ein Systemdiagramm der Hauptkomponenten der Chiffriervorrichtung (CF).

Fig. 2 ist ein Systemdiagramm der Komponenten der Chiffriervorrichtung CF, des Software-Treibers CFAP und verschiedener Anwendungsprogramme zur Chiffrierung.

In Fig. 3 ist dargestellt, wo Steuervektoren in verschiedenen systemübergreifenden Kommunikationskonstellationen angewendet werden.

In Fig. 4 ist die fundamentale Einteilung der Chiffrierschlüssel skizziert.

In Fig. 5 ist die Einteilung der Datenschlüssel dargestellt.

Fig. 6 zeigt die Einteilung der persönlichen Identifikationszahlen PIN.

Fig. 7 zeigt die Einteilung der Schlüsselchiffrierschlüssel.

Fig. 8 ist eine Zusammenfassung der relativen Prioritäten bei der Implementierung der verschiedenen Arten der Schlüsseleinteilung.

Fig. 9 enthält eine Zusammenfassung der Gliederung für jede der definierten Schlüsselarten.

Fig. 10 zeigt das allgemeine Format für Steuervektoren (CV).

In Fig. 11 ist das CV-Format für Vertraulichkeitsschlüssel dargestellt.

In Fig. 12 ist das Verfahren zur Chiffrierung eines Schlüssels K unter dem Hauptschlüssel mit einem erweiterten Steuervektor dargestellt.

In Fig. 13 ist das CV-Format für MAC-Schlüssel dargestellt.

In Fig. 14 ist das CV-Format für Daten/Kompatibilitätsschlüssel dargestellt.

In Fig. 15 ist das CV-Format für Daten/Umwandlungsschlüssel (XLT-Schlüssel) dargestellt.

■ In Fig. 16 ist das CV-Format für ANSI-Datenschlüssel dargestellt.

In Fig. 17 ist das CV-Format für PIN-Chiffrierschlüssel dargestellt.

In Fig. 18 ist das CV-Format für PIN-Generierungsschlüssel dargestellt.

In Fig. 19 ist das CV-Format für Schlüsselchiffrierschlüssel (KEK) des Senders dargestellt.

In Fig. 20 ist das CV-Format für KEK des Empfängers dargestellt.

In Fig. 21 ist das CV-Format für ANSI-KEKs dargestellt.

In Fig. 22 ist das CV-Format für Schlüsselteile dargestellt.

In Fig. 23 ist das CV-Format für Zwischen-ICVs dargestellt.

In Fig. 24 ist das CV-Format für Tokens dargestellt.

Fig. 25 ist ein Blockdiagramm der Codieranweisung.

Fig. 26 ist ein Blockdiagramm der Decodieranweisung.

Fig. 27 ist ein Blockdiagramm der Chiffrieranweisung.

Fig. 28 ist ein Blockdiagramm der Dechiffrieranweisung.

Fig. 29 ist ein Blockdiagramm der Anweisung "Meldungsidentifikations-Überprüfungscode generieren" (GMAC).

Fig. 30 ist ein Blockdiagramm der Anweisung "Meldungsidentifikations-Überprüfungscode überprüfen" (VMAC).

Fig. 31 ist ein Blockdiagramm der Anweisung "Chiffretext umwandeln".

In Fig. 32 sind die Arten und Unterarten von Schlüsseln aufgelistet, die für jeden Modus der Anweisung "Schlüsselsatz generieren" (GKS) generiert werden können.

Fig. 33 ist ein Blockdiagramm der Anweisung "Schlüsselsatz generieren".

In Fig. 34 sind die zulässigen Kombinationen von CV-Arten für das durch GKS OP-OP-Modus generierte Schlüsselpaar aufgeführt.

In Fig. 35 sind die zulässigen Kombinationen von linken und rechten CV-Attributen von in den verschiedenen Modi von GKS verwendeten Import- und Export-KEKs aufgeführt.

Fig. 36 ist eine Liste der zulässigen Kombinationen von CV-Schlüsselformen und Verbindungssteuerungen für Paare von durch verschiedene Modi von GKS generierten importierten oder exportierten Schlüsseln.

Fig. 37 ist eine Tabelle der zulässigen Kombinationen von CV-Arten für das durch GKS OP-IM-Modus generierte Schlüsselpaar.

Fig. 38 ist ein Blockdiagramm der Anweisung "Umchiffrieren vom Hauptschlüssel".

Fig. 39 ist ein Blockdiagramm der Anweisung "Umchiffrieren unter den Hauptschlüssel".

Fig. 40 ist ein Blockdiagramm der Anweisung "Schlüssel generieren".

Fig. 41 ist ein Blockdiagramm der Anweisung "Unter Hauptschlüssel chiffrieren".

Fig. 42 ist ein Blockdiagramm der Anweisung "Schlüssel umwandeln".

Fig. 43 ist eine Tabelle der zulässigen Kombinationen von linken bzw. rechten CV-Schlüsselformattributen für Import- und Export-KEKs, die in der Anweisung "Schlüssel umwandeln" verwendet werden.

Fig. 44 ist das Blockdiagramm für Modus 0 (Schlüsselmodus) der Anweisung "Umchiffrieren unter den neuen Hauptschlüssel".

Fig. 45 ist das Blockdiagramm für Modus 1 (Tokenmodus) der Anweisung "Umchiffrieren unter den neuen Hauptschlüssel".

Fig. 46 ist das Blockdiagramm für Modus 0 (Schlüsselmodus) der Anweisung "Umchiffrieren unter den aktuellen Hauptschlüssel".

Fig. 47 ist das Blockdiagramm für Modus 1 (Tokenmodus) der Anweisung "Umchiffrieren unter den aktuellen Hauptschlüssel".

Fig. 48 und Fig. 49 sind Blockdiagramme der Anweisung "ANSI Partiellen Beglaubigungsschlüssel erstellen".

Fig. 50 ist das Blockdiagramm für die Anweisung "ANSI Umchiffrieren vom Hauptschlüssel" (ARFMK).

In Fig. 51 sind die zulässigen CV-Verwendungsattribute für einen durch ARFMK zu exportierenden Datenschlüssel aufgeführt.

In Fig. 52 sind die zulässigen Kombinationen von CV-Art und Schlüsselform für die linke bzw. rechte Hälfte des Export-KEK und von den entsprechenden Attributen des in der Anweisung ARFMK zu exportierenden Schlüssels aufgeführt.

Fig. 53 ist eine Tabelle der zulässigen CV-Verwendungsattribute, die für den als Nebenprodukt von ARFMK erzeugten CSM MAC-Schlüssel angegeben werden können.

Fig. 54 ist das Blockdiagramm für die Anweisung "ANSI Umchiffrieren unter den Hauptschlüssel" (ARTMK).

In Fig. 55 sind die zulässigen CV-Verwendungsattribute für einen durch ARTMK zu importierenden Datenschlüssel aufgeführt.

In Fig. 56 sind die zulässigen Kombinationen von CV-Art und Schlüsselform für die linke bzw. rechte Hälfte des Import-KEK



und von den entsprechenden Attributen des in der Anweisung ARTMK zu importierenden Schlüssels aufgeführt.

Fig. 57 ist eine Tabelle der zulässigen CV-Verwendungsattribute, die für den als Nebenprodukt von ARTMK erzeugten CSM MAC-Schlüssel angegeben werden können.

Fig. 58 und Fig. 59 sind Blockdiagramme der Anweisung "ANSI Schlüssel umwandeln" (AXLTKEY).

Fig. 60 ist eine Tabelle der zulässigen CV-Verwendungsattribute, die für den als Nebenprodukt von AXLTKEY erzeugten CSM MAK-Schlüssel angegeben werden können.

Fig. 61 ist das Blockdiagramm der Anweisung "ANSI KDs kombinieren" (ACOMBKD).

Fig. 62 ist eine Tabelle der zulässigen CV-Verwendungsattribute für die erste von zwei "partiellen" CSM MAC-Schlüssel-Eingangsinformationen für die Anweisung ACOMBKD.

Fig. 63 ist eine Tabelle der zulässigen CV-Verwendungsattribute für die zweite von zwei "partiellen" CSM MAC-Schlüssel-Eingangsinformationen für die Anweisung ACOMBKD.

Fig. 64 ist eine Tabelle der zulässigen CV-Verwendungsattribute, die für den durch ACOMBKD erzeugten Schlüssel angegeben werden können.

Fig. 65, 66, 67 und 68 sind Tabellen üblicher Schlüsselverwaltungsanforderungen bzw. -aktivitäten und der Anweisungen, die dafür verwendet werden können.

Fig. 69 zeigt die Transformationen von Schlüsselstatus (OP, IM, EX), die von bestimmten Anweisungen durchgeführt werden. Export-

und Importkanäle sind Einbahnstraßen für die von der Chiffrier-  
vorrichtung importierten bzw. exportierten Schlüssel.

Fig. 70 ist eine Tabelle der zulässigen CV-Arten, die über die  
verschiedenen Kanalarten (CV, CV=0 und ANSI) importiert oder  
exportiert werden können.

Fig. 71 ist eine Tabelle der zulässigen Kombinationen von Ver-  
teilungsmodi und CV-Verbindungssteuerungsattributen.

Fig. 72 ist ein Schema des Schlüsselumwandlungsprozesses, der an  
Knoten C für Knoten A und B durchgeführt wird.

In Fig. 73 ist ein Teil der Schlüsselchiffrierschlüssel-Datei  
(KDDS), in der KEKs gespeichert sind.

Fig. 74 ist ein Diagramm eines ANSI X9.17-Systemknotens.

In Fig. 75 sind die funktionalen Schnittstellen zwischen den  
Komponenten in einem ANSI X9.17-Knoten dargestellt.

Fig. 76 ist ein Systemdiagramm einer ANSI Punkt-zu-Punkt-Verbin-  
dung.

Fig. 77 ist ein Systemdiagramm einer ANSI Schlüsselverteilungs-  
zentrum-Umgebung (KDC).

Fig. 78 ist ein Systemdiagramm einer ANSI Schlüsselumwandlungs-  
zentrum-Umgebung (KTC).

In Fig. 79 ist die Verteilung von KEKs ((\*KK) in einer CCA ANSI  
X9.17-Implementierung tabellarisch dargestellt.

In Fig. 80 und 81 ist die Verteilung von Datenschlüsseln (KD) in  
einer CCA ANSI X9.17-Implementierung tabellarisch dargestellt.

In Fig. 82 ist der ECB-Modus (elektronisches Codebuch) der DES-Chiffrierung dargestellt.

In Fig. 83 ist der CBC-Modus (Chiffreblockkettung) der DES-Chiffrierung dargestellt.

In Fig. 84 ist der CBC-Modus der DES-Dechiffrierung dargestellt.

In Fig. 85 ist der Algorithmus zur Meldungs-Identifikationsüberprüfung (MAC) dargestellt.

In Fig. 86, 87 und 88 sind die Gleichungen für alle Anweisungen aufgeführt.

Es werden ein Verfahren und eine Vorrichtung zur flexiblen Schlüsselerwendung beschrieben, die hier als Chiffrierarchitektur (CA) bezeichnet werden. Das Verfahren erzwingt eine strikte Schlüsseltrennung innerhalb der lokalen Chiffriervorrichtung und auf systemübergreifender Ebene (d.h. bei der Übertragung). Das Verfahren arbeitet mit Steuervektoren, die den Empfänger oder Benutzer eines Schlüssels zwingen, den Schlüssel so zu verwenden wie vom Urheber vorgesehen.

Alle außerhalb der Chiffriervorrichtung auf einem unsicheren Medium gespeicherten Schlüssel werden unter einem Schlüssel chiffriert, der aus einer Funktion des Hauptschlüssels und eines speziellen vom Urheber des gespeicherten Schlüssels vorgegebenen Steuervektorwertes gebildet wird. Der Hauptschlüssel und möglicherweise einige weitere Schlüssel sind unverschlüsselt innerhalb der physisch sicheren Chiffriervorrichtung gespeichert. Zu keiner Zeit erscheint ein Schlüssel unabsichtlich unverschlüsselt außerhalb der sicheren Chiffriervorrichtung.

Von einem System an ein anderes übertragene Schlüssel werden unter einem Schlüssel chiffriert, der durch eine Funktion aus einem Schlüsselchiffrierschlüssel und speziellen Steuervektor-

werten, die vom Urheber (und eventuell vom Sender) des Schlüssels vorgegeben wurden, gebildet wird. Wo eine solche Kontrolle der Schlüsselverwendung mit der Unterstützung für spezielle Schlüsselverteilungsprotokolle in Konflikt gerät, ist eine Steuerung bei der Verbindung nicht durchzusetzen.

Es wird eine Gruppe elementarer Chiffrierfunktionen oder Anweisungen definiert. Diese Anweisungen bilden die Basis der funktionalen Sicherheit. Jede Anweisung ist in Form von Ein- und Ausgabeparametern, Funktionsbeschreibung und Steuervektorverarbeitung definiert. Wegen der Integrität sind die Anweisungen nur zur ausschließlichen Verwendung innerhalb der sicheren Chiffriervorrichtung vorgesehen.

Ein sehr starkes und positives Merkmal des Chiffrierverfahrens ist die strikte Einhaltung des Prinzips der beschränkten Funktion. Dieses Prinzip fordert, daß die Chiffrierschnittstellen so implementiert werden, daß nur die gewünschten und vorausgesehenen Chiffrierfunktionen erlaubt sind, und nichts anderes. Genau dieses "nichts anderes" ist wichtig. Es hat sich gezeigt, daß Angriffe häufig entwickelt werden, indem die eingerichteten Schnittstellen früherer Systeme auf eine Weise verwendet wurden, die nicht speziell benötigt oder unter der Architektur nicht aufgerufen, aber durch die Implementierung nicht ausdrücklich verboten worden war.

In Fig. 1 ist ein Blockdiagramm des Datenverarbeitungssystems mit der darin enthaltenen Chiffriervorrichtung dargestellt. Das Datenverarbeitungssystem 2 führt ein Programm aus wie z.B. das Chiffriervorrichtungs-Zugriffsprogramm und die Anwendungsprogramme, die in Fig. 2 dargestellt sind. Diese Programme geben Chiffrierserviceanforderungen zur Verwaltung von Chiffrierschlüsseln, denen Steuervektoren zugeordnet sind, aus. Das allgemeine Format eines Steuervektors ist in Fig. 10 dargestellt. Steuervektoren definieren die Funktion, zu deren Ausführung der zugehörige Schlüssel von seinem Urheber ermächtigt wurde. Die

hier beschriebene Chiffrierarchitekturerfindung ist eine Vorrichtung und ein Verfahren zur Überprüfung, daß die Schlüsselverwaltungsfunktionen, die vom Programm für einen Chiffrierschlüssel angefordert werden, vom Urheber des Schlüssels autorisiert sind.

Wie in Fig. 1 zu sehen ist, ist im Datenverarbeitungssystem 2 eine durch eine sichere Grenze 6 gekennzeichnete Chiffriervorrichtung 4 enthalten oder daran angeschlossen. Durch die sichere Grenze 6 verläuft ein Ein-/Ausgangspfad 8, über den die Chiffrierserviceanforderung, Chiffrierschlüssel und die zugehörigen Steuervektoren vom Programm empfangen werden können. Der Ein-/Ausgangspfad 8 gibt Antworten auf solche Chiffrieranforderungen von der Chiffriervorrichtung aus. Innerhalb der sicheren Grenze 6 befindet sich ein Chiffrieranweisungsspeicher 10, der durch Einheiten des Busses 12 mit dem Ein-/Ausgangspfad 8 verbunden ist. An den Anweisungsspeicher 10 ist eine Steuervektor-Prüfeinheit 14 sowie eine Chiffrierverarbeitungseinheit 16 angeschlossen. An die Chiffrierverarbeitungseinheit 16 ist ein Hauptschlüsselspeicher 18 angeschlossen. Die Chiffriervorrichtung 4 bietet einen sicheren Ort zur Ausführung von Schlüsselverwaltungsfunktionen als Antwort auf die empfangene Serviceanforderung.

- ☐ Der Chiffrieranweisungsspeicher 10 empfängt über den Ein-/Ausgangspfad 8 eine Chiffrierserviceanforderung zur Ausführung einer Schlüsselverwaltungsfunktion mit einem Chiffrierschlüssel. Die Steuervektor-Prüfeinheit 14 besitzt einen mit dem Ein-/Ausgangspfad 8 verbundenen Eingang zum Empfang eines dem Chiffrierschlüssel zugeordneten Steuervektors. Sie besitzt ferner einen an den Chiffrieranweisungsspeicher 10 angeschlossenen Eingang zum Empfang von Steuersignalen, die die Prüfung, ob der Steuervektor die von der Chiffrierserviceanforderung angeforderte Schlüsselverwaltungsfunktion autorisiert, startet.

Die Steuervektor-Prüfeinheit 14 besitzt einen an einen Eingang der Chiffrierverarbeitungseinheit 16 angeschlossenen Autorisierungsausgang 20, über den signalisiert wird, daß die Schlüsselverwaltungsfunktion autorisiert ist. Der Empfang des Autorisierungssignals durch die Chiffrierverarbeitungseinheit 16 startet die Ausführung der angeforderten Schlüsselverwaltungsfunktion mit dem Chiffrierschlüssel. An die Chiffriervorrichtung 14 ist über den Ein-/Ausgangspfad ein Chiffrierschlüsselspeicher 22 angeschlossen. Dieser speichert den Chiffrierschlüssel in einer verschlüsselten Form, in der der Chiffrierschlüssel unter einem Speicherschlüssel chiffriert ist, der ein logisches Produkt aus dem zugehörigen Steuervektor und dem im Hauptschlüsselspeicher 18 gespeicherten Hauptschlüssel ist.

Ein Beispiel für die Dechiffrierung eines chiffrierten Schlüssels aus dem Chiffrierschlüsselspeicher 22 findet statt, wenn der Chiffrieranweisungsspeicher 10 über den Ein-/Ausgangspfad eine Chiffrierserviceanforderung zur Dechiffrierung des Chiffrierschlüssels aus dem Chiffrierschlüsselspeicher 22 empfängt. Die Steuervektor-Prüfeinheit 14 sendet daraufhin auf Leitung 20 ein Autorisierungssignal an die Chiffrierverarbeitungseinheit 16, das besagt, daß die Funktion zur Dechiffrierung des Chiffrierschlüssels autorisiert ist. Die Chiffrierverarbeitungseinheit 16 empfängt dann als Antwort auf das Autorisierungssignal die verschlüsselte Form des Chiffrierschlüssels vom Chiffrierschlüsselspeicher 22 und dechiffriert diese unter dem Speicherschlüssel, der ein logisches Produkt aus dem zugehörigen Steuervektor und dem im Hauptschlüsselspeicher 18 gespeicherten Hauptschlüssel ist.

Der Speicherschlüssel ist ein EXKLUSIV-ODER-Produkt des zugehörigen Steuervektors und des im Hauptschlüsselspeicher 18 gespeicherten Hauptschlüssels. Daß das logische Produkt im bevorzugten Ausführungsbeispiel eine EXKLUSIV-ODER-Operation ist, schließt andere logische Operationen jedoch nicht aus.

Der zugehörige Steuervektor, dessen allgemeines Format in Fig. 10 dargestellt ist, wird mit der chiffrierten Form des betreffenden Chiffrierschlüssels im Chiffrierschlüsselspeicher 22 gespeichert. Da alle Schlüssel unter dem Hauptschlüssel verschlüsselt im Chiffrierschlüsselspeicher gespeichert werden, kann ein konstantes Verfahren zur Chiffrierung und Dechiffrierung von darunter verschlüsselten Schlüsseln angewendet werden.

Der Steuervektor, dessen allgemeines Format in Fig. 10 dargestellt ist, enthält Felder, in denen die autorisierten Arten von Chiffrierfunktionen einschließlich Schlüsselverwaltungsfunktionen, Datenchiffrier- und Datendechiffrierfunktionen und Funktionen zur Verarbeitung der persönlichen Identifikationsnummer (PIN) definiert sind. Bei den Schlüsselverwaltungsanwendungen ist die Art der Schlüsselverwaltungsfunktionen im Feld "CV-ART" angegeben. Der Steuervektor enthält noch weitere Felder, in denen die Exportsteuerung für die Schlüssel und zugehörige verschlüsselte Daten sowie die Verwendung der Schlüssel und der zugehörigen verschlüsselten Daten festgelegt werden kann.

Die Anordnung in Fig. 1 kann einen Schlüsselsatz generieren, d.h. ein Schlüsselpaar mit mehreren Verwendungsklassen. Die Chiffriererverarbeitungseinheit besitzt einen Eingang, der über den Bus 12 an eine Zufallszahlenquelle 26 oder eine gespeicherte Zufallszahl aus dem Arbeitsspeicher 24 der Verschlüsselungsvorrichtung 4 angeschlossen ist, und über den sie eine Zufallszahl empfangen kann.

Der Chiffrieranweisungsspeicher 10 empfängt daraufhin über den Ein-/Ausgangspfad 8 eine Chiffrierserviceanforderung zur Generierung eines Schlüsselpaares aus der Zufallszahlenausgabe der Zufallszahlenquelle 26 mit dem zugehörigen ersten und zweiten Steuervektor C3 und C4. Die Steuervektor-Prüfeinheit 14 sendet dann als Antwort darauf auf Leitung 20 ein Autorisierungssignal an die Chiffriererverarbeitungseinheit 16, das besagt, daß die Funktion zur Generierung eines Schlüsselpaares aus der Zufalls-

zahl autorisiert ist. Die Chiffrierverarbeitungseinheit 16 gibt dann als Antwort auf das Autorisierungssignal auf Leitung 20 die Zufallszahl als ersten generierten Schlüssel in einer verschlüsselten Form aus, in der die erste Zufallszahl unter einem Schlüssel chiffriert ist, der ein logisches Produkt aus dem ersten Steuervektor C3 und einem ersten Schlüssel K1 ist. Die Chiffrierverarbeitungseinheit 16 gibt dann als Antwort auf das Autorisierungssignal auf Leitung 20 außerdem die Zufallszahl als zweiten generierten Schlüssel in einer verschlüsselten Form aus, in der die Zufallszahl unter einem Schlüssel chiffriert ist, der ein logisches Produkt aus dem zweiten Steuervektor C4 und einem zweiten Schlüssel K2 ist. Durch die Steuervektoren C3 und C4 können mehrere Verwendungszweckkombinationen autorisiert werden. Eine Kombination gilt für die Erzeugung von zwei Schlüsseln, die im lokalen Datenverarbeitungssystem 2 benutzt werden können; in diesem Fall bestehen der erste Schlüssel K1 und der zweite Schlüssel K2 aus der Zufallszahl, und der erste Steuervektor C3 und der zweite Steuervektor C4 autorisieren nur die lokale Verwendung innerhalb des lokalen Datenverarbeitungssystems 2. Eine zweite Kombination gilt beispielsweise für die Erzeugung eines ersten generierten Schlüssels, der nur im lokalen Datenverarbeitungssystem 2 verwendet werden kann, und eines zweiten generierten Schlüssels, der an ein an das lokale System 2 angeschlossenes fernes Datenverarbeitungssystem 30 exportiert werden kann. In diesem Fall ist der erste Schlüssel K1 die Zufallszahl, und der erste Steuervektor C3 autorisiert nur die lokale Verwendung innerhalb des lokalen Datenverarbeitungssystems 2, wohingegen der zweite Schlüssel K2 ein Schlüsselchiffrierschlüssel KEK2 ist und der zweite Steuervektor C4 den Export an das ferne Datenverarbeitungssystem 30 autorisiert. Ein drittes Beispiel ist die Erzeugung eines ersten generierten Schlüssels, der an ein erstes an das lokale Datenverarbeitungssystem 2 angeschlossenes fernes Datenverarbeitungssystem 30 exportiert werden kann, und eines zweiten generierten Schlüssels, der an ein zweites an das lokale Datenverarbeitungssystem 2 angeschlossenes fernes Datenverarbeitungssystem 30' exportiert werden kann. In diesem Fall ist der



erste Schlüssel K1 ein Schlüsselchiffrierschlüssel KEK1, und der erste Steuervektor C3 autorisiert den Export an das erste ferne Datenverarbeitungssystem 30; der zweite Schlüssel K2 ist ein Schlüsselchiffrierschlüssel KEK2, und der zweite Steuervektor C4 autorisiert den Export an das zweite ferne Datenverarbeitungssystem 30'. Eine vierte Kombinationsmöglichkeit ist Verwendung zur Erzeugung eines ersten generierten Schlüssels, der nur im lokalen Datenverarbeitungssystem 2 benutzt werden kann, und eines zweiten generierten Schlüssels, der von einem an das lokale Datenverarbeitungssystem 2 angeschlossenen fernen Datenverarbeitungssystem 30 importiert werden kann. In diesem Fall ist der erste Schlüssel K1 die Zufallszahl, und der erste Steuervektor C3 autorisiert nur die normale Verwendung innerhalb des lokalen Datenverarbeitungssystems 2; der zweite Schlüssel K2 ist ein Schlüsselchiffrierschlüssel KEK2, und der zweite Steuervektor C4 autorisiert den Import vom fernen Datenverarbeitungssystem 30 in das lokale Datenverarbeitungssystem 2. Ein fünftes Fallbeispiel ist die Verwendung zur Erzeugung eines ersten generierten Schlüssels, der von einem ersten an das lokale Datenverarbeitungssystem 2 angeschlossenen fernen Datenverarbeitungssystem 30 importiert werden kann, und eines zweiten generierten Schlüssels, der an ein zweites an das lokale Datenverarbeitungssystem 2 angeschlossenes fernes Datenverarbeitungssystem 30' exportiert werden kann. In diesem Fall ist der erste Schlüssel K1 ein Schlüsselchiffrierschlüssel KEK1, und ein erster Steuervektor C3 autorisiert den Import vom ersten fernen Datenverarbeitungssystem 30; der zweite Schlüssel K2 ist ein Schlüsselchiffrierschlüssel KEK2, und ein zweiter Steuervektor C4 autorisiert den Export an das zweite ferne Datenverarbeitungssystem 30'.

Die in Fig. 1 dargestellte Anordnung kann die Operation "Umchiffrieren vom Hauptschlüssel" folgendermaßen ausführen. Das Datenverarbeitungssystem 2 ist über eine DFV-Verbindung 28 mit einem fernen Datenverarbeitungssystem 30 verbunden, das wie das Datenverarbeitungssystem 2 über einen geheimen Schlüsselchiffrierschlüssel KEK verfügt. Im Chiffrierschlüsselspeicher 22 wird

der Schlüsselchiffrierschlüssel KEK in einer verschlüsselten Form gespeichert, in der KEK unter einem Speicherschlüssel chiffriert ist, der ein logisches Produkt aus einem zugehörigen Steuervektor C1 und dem Hauptschlüssel ist. Der Chiffrierschlüsselspeicher 22 enthält auch den Chiffrierschlüssel als Schlüssel K in einer verschlüsselten Form, in der K unter einem Speicherschlüssel chiffriert ist, der ein logisches Produkt aus einem zugehörigen Steuervektor C2 und dem Hauptschlüssel ist.

Der Chiffrieranweisungsspeicher 10 empfängt über den Ein-/Ausgangspfad 8 eine Chiffrierserviceanforderung zur Umchiffrierung des Chiffrierschlüssels K vom Hauptschlüssel, die unter dem Schlüsselchiffrierschlüssel KEK chiffriert werden soll, um zusammen mit einem Steuervektor C3 an das ferne Datenverarbeitungssystem 30 exportiert zu werden. Die Steuervektor-Prüfeinheit 14 sendet daraufhin auf Leitung 20 ein Autorisierungssignal an die Chiffrierverarbeitungseinheit 16, das besagt, daß die Funktion der Wiederverschlüsselung des Chiffrierschlüssels K vom Hauptschlüssel unter den Schlüsselchiffrierschlüssel KEK zu Exportzwecken autorisiert ist.

Die Chiffrierverarbeitungseinheit 16 empfängt als Antwort auf das Autorisierungssignal auf Leitung 20 die chiffrierte Form des Chiffrierschlüssels K vom Chiffrierschlüsselspeicher 22 und dechiffriert diese chiffrierte Form unter einem Speicherschlüssel, der ein logisches Produkt aus dem zugehörigen Steuervektor C2 und dem Hauptschlüssel ist. Ferner empfängt sie als Antwort auf das Autorisierungssignal auf Leitung 20 die chiffrierte Form des Schlüsselchiffrierschlüssels KEK vom Chiffrierschlüsselspeicher 22 und dechiffriert diese chiffrierte Form unter einem Speicherschlüssel, der ein logisches Produkt aus dem zugehörigen Steuervektor C1 und dem Hauptschlüssel ist.

Anschließend führt die Chiffrierverarbeitungseinheit 16 als Antwort auf das Autorisierungssignal auf Leitung 20 die Umchiffrierung des Chiffrierschlüssels K unter einem logischen Produkt aus

dem zugehörigen Steuervektor C3 und dem Schlüsselchiffrierschlüssel KEK durch und sendet den umchiffrierten Chiffrierschlüssel K zusammen mit dem zugehörigen Steuervektor C3 über die DFV-Verbindung 28 an das erste ferne Datenverarbeitungssystem 30.

Die in Fig. 1 dargestellte Anordnung kann die Operation "Umchiffrieren unter den Hauptschlüssel" folgendermaßen ausführen. Das Datenverarbeitungssystem 2 ist über eine DFV-Verbindung 28 mit einem fernen Datenverarbeitungssystem 30 verbunden, das wie das Datenverarbeitungssystem 2 über einen geheimen Schlüsselchiffrierschlüssel KEK verfügt. Im Chiffrierschlüsselspeicher 22 wird der Schlüsselchiffrierschlüssel KEK in einer verschlüsselten Form gespeichert, in der KEK unter einem Speicherschlüssel chiffriert ist, der ein logisches Produkt aus einem zugehörigen Steuervektor C1 und dem Hauptschlüssel ist.

Das ferne Datenverarbeitungssystem 30 überträgt über die DFV-Verbindung 28 einen unter dem Schlüsselchiffrierschlüssel KEK chiffrierten Chiffrierschlüssel K zusammen mit einem Steuervektor C3 an das lokale Datenverarbeitungssystem 2. Der Chiffrieranweisungsspeicher 10 empfängt über den Ein-/Ausgangspfad 8 eine Chiffrierserviceanforderung zur Umchiffrierung des Chiffrierschlüssels K vom Chiffrierschlüssel KEK, der unter dem Hauptschlüssel chiffriert werden soll, um zusammen mit einem Steuervektor C2 im Chiffrierschlüsselspeicher 22 abgespeichert zu werden. Die Steuervektor-Prüfeinheit 14 sendet daraufhin auf Leitung 20 ein Autorisierungssignal an die Chiffrierverarbeitungseinheit 16, das besagt, daß die Funktion der Umchiffrierung des Chiffrierschlüssels K vom Schlüsselchiffrierschlüssel KEK unter den Hauptschlüssel zur Speicherung autorisiert ist. Die Chiffrierverarbeitungseinheit 16 dechiffriert dann als Antwort auf das Autorisierungssignal auf Leitung 20 den Schlüssel vom Schlüsselchiffrierschlüssel KEK, verschlüsselt ihn zusammen mit dem Steuervektor C2 unter dem Hauptschlüssel und übergibt den umchiffrierten Schlüssel K an den Chiffrierschlüsselspeicher 22.

Mit der Anordnung aus Fig. 1 kann ferner wie folgt ein einzelner Schlüssel generiert werden. Die Zufallszahlenquelle 26 besitzt einen Ausgang, der über den Bus 12 an die Chiffrierverarbeitungseinheit 16 angeschlossen ist. Über diesen Ausgang wird eine Zufallszahl an die Chiffrierverarbeitungseinheit 16 gesendet. Der Chiffrieranweisungsspeicher 10 empfängt über den Ein-/Ausgangspfad 8 eine Chiffrierserviceanforderung zur Generierung eines Schlüssels aus der Zufallszahl und einem zugehörigen Steuervektor C1. Die Steuervektor-Prüfeinheit 14 sendet daraufhin auf Leitung 20 ein Autorisierungssignal an die Chiffrierverarbeitungseinheit 16, das besagt, daß die Funktion der Generierung eines Schlüssels aus Zufallszahlen autorisiert ist. Als Antwort auf das Autorisierungssignal auf Leitung 20 sendet die Chiffrierverarbeitungseinheit 16 die Zufallszahl als generierten Schlüssel in einer verschlüsselten Form, in der die Zufallszahl unter einem Schlüssel chiffriert ist, der ein logisches Produkt aus dem zugehörigen Steuervektor C1 und dem Hauptschlüssel ist.

Die Anordnung aus Fig. 1 kann ferner wie folgt eine Schlüsselumwandlungsfunktion ausführen. Das Datenverarbeitungssystem 2 aus Fig. 1 ist ein lokales System, das über eine DFV-Verbindung 28 mit einem ersten fernen Datenverarbeitungssystem 30 verbunden ist, mit dem es einen geheimen Schlüsselchiffrierschlüssel KEK1 für den Import gemeinsam hat. Das lokale Datenverarbeitungssystem 2 ist außerdem über eine DFV-Verbindung 28' mit einem zweiten fernen Datenverarbeitungssystem 30' verbunden, mit dem es einen geheimen Schlüsselchiffrierschlüssel KEK2 für den Export gemeinsam hat. Im Chiffrierschlüsselspeicher 22 wird der Import-Schlüsselchiffrierschlüssel KEK1 in einer verschlüsselten Form gespeichert, in der KEK1 unter einem Speicherschlüssel chiffriert ist, der ein logisches Produkt aus einem zugehörigen Steuervektor C1 und dem Hauptschlüssel ist. Im Chiffrierschlüsselspeicher 22 wird der Export-Schlüsselchiffrierschlüssel KEK2 in einer verschlüsselten Form gespeichert, in der KEK2 unter einem Speicherschlüssel chiffriert ist, der ein logisches Produkt aus einem zugehörigen Steuervektor C2 und dem Hauptschlüssel ist.

Das erste ferne Datenverarbeitungssystem 30 überträgt über die DFV-Verbindung 28 einen unter dem Import-Schlüsselchiffrierschlüssel KEK1 chiffrierten Chiffrierschlüssel K zusammen mit einem Steuervektor C3 an das lokale Datenverarbeitungssystem 2. Der Chiffrieranweisungsspeicher 10 empfängt über den Ein-/Ausgangspfad 8 eine Chiffrierserviceanforderung zur Umwandlung des Chiffrierschlüssels K von der Chiffrierung unter dem Import-Schlüsselchiffrierschlüssel KEK1 unter den Export-Schlüsselchiffrierschlüssel KEK2, zusammen mit einem zugehörigen Steuervektor C3 zur Übertragung über die DFV-Verbindung 28' an das zweite Datenverarbeitungssystem 30'. Die Steuervektor-Prüfeinheit 14 sendet daraufhin auf Leitung 20 ein Autorisierungssignal an die Chiffrierverarbeitungseinheit 16, das besagt, daß die Funktion der Umwandlung des Chiffrierschlüssels K von der Chiffrierung unter dem Import-Schlüsselchiffrierschlüssel KEK1 in die Chiffrierung unter dem Export-Schlüsselchiffrierschlüssel KEK2 autorisiert ist.

Die Chiffrierverarbeitungseinheit 16 wandelt dann als Antwort auf das Autorisierungssignal auf Leitung 20 den Chiffrierschlüssel K zusammen mit dem Steuervektor C3 von der Chiffrierung unter dem Import-Schlüsselchiffrierschlüssel KEK1 in die Chiffrierung unter dem Export-Schlüsselchiffrierschlüssel KEK2 um, um ihn über die DFV-Verbindung 28' an das zweite Datenverarbeitungssystem 30' zu senden.

Mit den Steuervektoren C1, C2 und C3 können dabei verschiedene Einschränkungen festgelegt werden, z.B. um zu verhindern, daß vom lokalen Datenverarbeitungssystem 2 aus der Schlüssel K gelesen oder benutzt wird, so daß er sicher vom ersten Datenverarbeitungssystem 30 an das zweite Datenverarbeitungssystem 30' gesendet werden kann.

Die Anordnung aus Fig. 1 kann wie folgt eine Umchiffrierung vom aktuellen Hauptschlüssel unter einen neuen Hauptschlüssel durchführen. In einem Speicher für den aktuellen Hauptschlüssel, z.B.

im Arbeitsspeicher 24, wird ein aktueller Hauptschlüssel gespeichert, und im Hauptschlüsselspeicher 18 kann ein neuer Hauptschlüssel gespeichert werden. Sowohl der Speicher 24 für den aktuellen Hauptschlüssel als auch der Hauptschlüsselspeicher 18 sind mit der Chiffrierverarbeitungseinheit 16 und der Chiffrier-  
vorrichtung 4 verbunden. Der Chiffrierschlüsselspeicher 22 enthält einen Chiffrierschlüssel wie z.B. den Schlüssel K in einer chiffrierten Form, in der der Schlüssel K unter einem Speicherschlüssel chiffriert ist, der ein logisches Produkt aus einem zugeordneten Steuervektor C1 und dem aktuellen Hauptschlüssel ist. Der Chiffrieranweisungsspeicher empfängt über den Ein-/Ausgangspfad 8 eine Chiffrierserviceanforderung, daß der Chiffrierschlüssel K mit dem zugehörigen Steuervektor C1 vom aktuellen Hauptschlüssel unter den neuen Hauptschlüssel umchiffriert werden soll, und die Steuervektor-Prüfeinheit sendet daraufhin auf Leitung 20 ein Autorisierungssignal an die Chiffrierverarbeitungseinheit 16, das besagt, daß die Funktion zur Umchiffrierung des Chiffrierschlüssels K vom aktuellen Hauptschlüssel in den neuen Hauptschlüssel autorisiert ist.

Die Chiffrierverarbeitungseinheit 16 empfängt als Antwort auf das Autorisierungssignal auf Leitung 20 die chiffrierte Form des Chiffrierschlüssels K vom Chiffrierspeicher 22 und dechiffriert die dechiffrierte Form dieses Schlüssels unter einem Speicherschlüssel, der ein logisches Produkt aus dem zugehörigen Steuervektor C1 und dem aktuellen Hauptschlüssel ist. Dann chiffriert die Chiffrierverarbeitungseinheit den Chiffrierschlüssel K als Antwort auf das Autorisierungssignal auf Leitung 20 unter einem logischen Produkt aus einem zugehörigen Steuervektor C1 und dem neuen Hauptschlüssel neu und sendet den neu chiffrierten Chiffrierschlüssel K zusammen mit dem zugehörigen Steuervektor C1 an den Chiffrierschlüsselspeicher 22.

Mit der Anordnung aus Fig. 1 ist auch eine Reduzierung der durch einen Steuervektor festgelegten Nutzungsberechtigung eines Schlüssels möglich. Dazu wird im Chiffrierschlüsselspeicher 22

der Chiffrierschlüssel K in einer chiffrierten Form gespeichert, in der der Schlüssel K unter einem Speicherschlüssel chiffriert ist, der ein logisches Produkt aus einem zugehörigen Steuervektor C1 und dem Hauptschlüssel ist. Der Steuervektor C1 enthält ein Feld, in dem beispielsweise die Exportkontrolle festgelegt ist. Der Chiffrieranweisungsspeicher 10 empfängt über den Ein-/Ausgangspfad 8 eine Chiffrierserviceanforderung zur Herabsetzung der Steuervektorberechtigung des Steuervektors C1, und die Steuervektor-Prüfeinheit 14 sendet daraufhin auf Leitung 20 ein Autorisierungssignal an die Chiffrierverarbeitungseinheit 16, das besagt, daß die Funktion zur Reduzierung der Steuervektorberechtigung des Steuervektors C1 autorisiert ist.

Die Chiffrierverarbeitungseinheit 16 empfängt dann als Antwort auf das Autorisierungssignal auf Leitung 20 die chiffrierte Form des Chiffrierschlüssels K aus dem Chiffrierschlüsselspeicher 22 und chiffriert die dechiffrierte Form davon unter einem Speicherschlüssel, der ein logisches Produkt aus dem zugehörigen Steuervektor C1 und dem Hauptschlüssel ist. Die Chiffrierverarbeitungseinheit 16 ersetzt dann als Antwort auf das Autorisierungssignal auf Leitung 20 den Steuervektor C1 durch einen zweiten Steuervektor C2, in dessen Exportkontrollfeld eine niedrigere Berechtigung angegeben ist als beim vorigen Steuervektor C1. Dann chiffriert die Chiffrierverarbeitungseinheit 16 als Antwort auf das Autorisierungssignal auf Leitung 20 den Schlüssel K mit dem zweiten Steuervektor C2 unter dem Hauptschlüssel und sendet den chiffrierten Schlüssel K zusammen mit dem zweiten Steuervektor C2 an den Chiffrierspeicher 22.

Mit der Anordnung aus Fig. 1 wird eine Umchiffrierung vom Hauptschlüssel mit gleichzeitiger Herabsetzung der Exportberechtigung für den Empfänger folgendermaßen bewerkstelligt. Das Datenverarbeitungssystem ist ein lokales Datenverarbeitungssystem 2, das mit einem ersten fernen Datenverarbeitungssystem 30 verbunden ist, mit dem es einen geheimen Schlüsselchiffrierschlüssel teilt. Im Chiffrierspeicher 22 ist der Schlüsselchiffrierschlüssel

sel KEK in einer chiffrierten Form gespeichert, in der der KEK unter einem Speicherschlüssel gespeichert ist, der ein logisches Produkt aus einem zugehörigen Steuervektor C1 und dem Hauptschlüssel ist. Der Chiffrierschlüsselspeicher 22 enthält den Chiffrierschlüssel als Schlüssel K in einer chiffrierten Form, in der er unter einem Speicherschlüssel chiffriert ist, der ein logisches Produkt aus einem zugehörigen Steuervektor C2 und dem Hauptschlüssel ist, wobei der zugehörige Steuervektor C2 ein Exportfeld besitzt, in dem eine erste Exportberechtigung festgelegt ist. Der Chiffrieranweisungsspeicher 10 empfängt über den Ein-/Ausgangspfad 8 eine Chiffrierserviceanforderung zur Umchiffrierung des Chiffrierschlüssels K vom Hauptschlüssel unter den Schlüsselchiffrierschlüssel KEK zum Export mit einem zugehörigen Steuervektor C3 an das erste ferne Datenverarbeitungssystem 30, wobei im Exportfeld des zugehörigen Steuervektors C3 eine zweite Exportberechtigung angegeben ist, die niedriger ist als die erste Exportberechtigung des Steuervektors C2.

Die Steuervektor-Prüfeinheit 14 sendet daraufhin auf Leitung 20 ein Autorisierungssignal an die Chiffrierverarbeitungseinheit 16, das besagt, daß die Funktion zur Umchiffrierung des Chiffrierschlüssels K vom Hauptschlüssel unter den Schlüsselchiffrierschlüssel KEK für den Export autorisiert ist. Die Chiffrierverarbeitungseinheit 16 empfängt als Antwort auf das Autorisierungssignal auf Leitung 20 die chiffrierte Form des chiffrierten Schlüssels K aus dem Chiffrierschlüsselspeicher 22 und dechiffriert die chiffrierte Form davon unter einem Speicherschlüssel, der ein logisches Produkt aus dem zugehörigen Steuervektor C2 und dem Hauptschlüssel ist. Die Chiffrierverarbeitungseinheit 16 empfängt als Antwort auf das Autorisierungssignal auf Leitung 20 außerdem die chiffrierte Form des Schlüsselchiffrierschlüssels KEK aus dem Chiffrierschlüsselspeicher 22 und dechiffriert die chiffrierte Form davon unter einem Speicherschlüssel, der ein logisches Produkt aus dem zugehörigen Steuervektor C1 und dem Hauptschlüssel ist.



Die Chiffrierverarbeitungseinheit 16 chiffriert dann als Antwort auf das Autorisierungssignal auf Leitung 20 den Chiffrierschlüssel K unter einem logischen Produkt aus dem zugehörigen Steuervektor C3 und dem Schlüsselchiffrierschlüssel KEK neu und gibt den umchiffrierten Chiffrierschlüssel K zusammen mit dem zugehörigen Steuervektor C3 aus, damit beide an das erste ferne Datenverarbeitungssystem 30 gesendet werden können. Das erste ferne Datenverarbeitungssystem hat nun eine niedrigere Berechtigung für den Wiederexport des Chiffrierschlüssels K, da im Exportfeld des zugehörigen Steuervektors C3 eine niedrigere Berechtigung angegeben ist.

Die Anordnung aus Fig. 1 kann ferner Operationen zur Verbindungssteuerung bereitstellen. Der Steuervektor enthält ein Feld zur Definition der Verbindungssteuerung, in dem angegeben ist, ob der dem Chiffrierschlüssel zugeordnete Steuervektor bei der Übertragung von lokalen Datenverarbeitungssystem an ein daran angeschlossenes fernes Datenverarbeitungssystem 30 aufgrund der Merkmale des fernen Datenverarbeitungssystems 30 nicht mitgesendet werden soll. Es ist beispielsweise möglich, daß das ferne Datenverarbeitungssystem nicht in der Lage ist, Steuervektoren aufzunehmen und zu verarbeiten. Die Steuervektoren, die bei der Übertragung vom zugehörigen Chiffrierschlüssel abgetrennt werden können, machen die Operationen mit dem betreffenden Chiffrierschlüssel weniger vertrauenswürdig. Solche Systeme wie das lokale Datenverarbeitungssystem 2, die mit Steuervektoren arbeiten können, sind deshalb sicherer als dies bei fernen Systemen möglich ist, die nicht über diese Verarbeitungsmöglichkeit verfügen. Der Steuervektor kann auch ein Feld enthalten, in dem angegeben ist, ob der zugehörige Schlüssel von einem ANSI-Datenverarbeitungssystem wie dem fernen Datenverarbeitungssystem 30 verarbeitet werden kann.

Bei der Anordnung aus Fig. 1 verfügt der Steuervektor, dessen allgemeines Format in Fig. 10 dargestellt ist, Felder, in denen die Trennung von Schlüsselchiffrierschlüsseln für sich gegensei-

tig ausschließende Verwendungszwecke erzwungen wird. In Fig. 7 ist dargestellt, wie die verschiedenen Schlüsselchiffrierschlüssel nach ihrem Verwendungszweck klassifiziert werden. Einander ausschließende Verwendungszwecke haben zum Beispiel Beglaubigungsschlüssel und nicht beglaubigende Schlüssel. Eine andere Form einander ausschließender Verwendungszwecke wären ein erster Schlüsselchiffrierschlüssel, der Steuervektoren verwendet, und ein zweiter Schlüsselchiffrierschlüssel, der nicht mit Steuervektoren arbeitet. Ein weiteres Beispiel für einander ausschließende Verwendungszwecke sind eine erste Art von Schlüsselchiffrierschlüsseln, die nur von Sendern verwendet werden können, und eine zweite Art von Schlüsselchiffrierschlüsseln, die nur von Empfängern verwendet werden können. Noch ein Beispiel für einander ausschließende Verwendungszwecke, bei denen Steuervektoren eine Schlüsseltrennung erzwingen, sind eine erste Art von Schlüsselchiffrierschlüssel, die zur Generierung von Schlüssel-paaren und zur Umwandlung von Schlüsseln für die Versendung verwendet werden können, ohne den Export vorhandener, unter einem Hauptschlüssel gespeicherter Datenschlüssel zu ermöglichen, und eine zweite Art von Schlüsselchiffrierschlüsseln, die zur Generierung von Schlüsselpaaren und zur Umwandlung von Schlüsseln für die Versendung verwendet werden können, aber den Export vorhandener, unter einem Hauptschlüssel gespeicherter Datenschlüssel erlauben. Ein anderes Beispiel einander ausschließender Verwendungszwecke, für die der Steuervektor eine Schlüsseltrennung erzwingt, sind eine erste Art von Schlüsselchiffrierschlüsseln, die nur für den Export generiert werden können, und eine zweite Art von Schlüsselchiffrierschlüsseln, die für die lokale Verwendung und für den Export generiert werden können. Ein weiteres Beispiel für einander ausschließende Verwendungszwecke, für die der Steuervektor eine Schlüsseltrennung erzwingt, sind eine erste Art von Schlüsselchiffrierschlüsseln, die zur Umwandlung, aber nicht für gewöhnliche lokale Operationen verwendet werden kann, und eine zweite Art von Schlüsselchiffrierschlüsseln, die sowohl zur Umwandlung als auch für gewöhnliche lokale Operationen verwendet werden kann. Ein weiteres Beispiel für einander

z

ausschließende Verwendungszwecke, für die der Steuervektor eine Schlüsseltrennung erzwingt, sind schließlich eine erste Art von Schlüsselchiffrierschlüsseln, die zur Umchiffrierung vom Hauptschlüssel verwendet werden kann, und eine zweite Art von Schlüsselchiffrierschlüsseln, die nicht zur Umchiffrierung vom Hauptschlüssel benutzt werden kann.

Der Steuervektor, dessen allgemeines Format in Fig. 10 dargestellt ist, kann außerdem ein Feld enthalten, in dem angegeben ist, ob der zugehörige Chiffrierschlüssel ein Schlüssel mit einfacher oder mit doppelter Länge ist.

Im folgenden werden die einzelnen Komponenten und Operationen der Anordnung ausführlicher erläutert.

In Fig. 2 sind die Hauptkomponenten eines Chiffriersubsystems dargestellt. Das Chiffriersubsystem besteht aus einer Chiffriervorrichtung (CF), dem Chiffriervorrichtungs-Zugriffsprogramm (CFAP) und dem Anwendungsprogramm (AP). Die CF ist gewöhnlich in einem physisch sicheren Gehäuse hardwaremäßig implementiert. Je nach Implementierung können sich auch die CF, das CFAP und das AP in einem physisch sicheren Kasten befinden.

Die Chiffriervorrichtung 4 besteht aus folgenden Komponenten:

- # Schlüsselregister - Die Register und ihre Verwendung werden nachstehend beschrieben:
- Hauptschlüsselregister 18 - Das Hauptschlüsselregister ist 128 Bit groß und enthält den Hauptschlüssel.
- Register für den neuen Hauptschlüssel (NMK) - Das Register für den neuen Hauptschlüssel ist 128 Bit groß und enthält den neuen Hauptschlüssel, der zum aktuellen Hauptschlüssel werden soll. Der neue Hauptschlüssel wird erst durch eine spezielle Anweisung zum aktuellen Hauptschlüssel; um den

Wert des neuen Hauptschlüssels in das Hauptschlüsselregister zu laden, wird die Anweisung SMK erteilt.

- Register für den alten Hauptschlüssel - Das Register für den alten Hauptschlüssel ist 128 Bit groß und enthält den Hauptschlüssel, der durch den neuen Hauptschlüssel ersetzt wurde. Der Mechanismus zur Aktualisierung des Hauptschlüssels ist so aufgebaut, daß zuerst der aktuelle Hauptschlüssel zum alten Hauptschlüssel wird, bevor der neue Hauptschlüssel zum aktuellen Hauptschlüssel wird.
- Schlüsselteileregister - Das Schlüsselteileregister ist 128 Bit groß und enthält den Wert eines Schlüsselteils (einer Schlüsselkomponente) oder einen vollständigen Schlüssel, der über eine Schlüsselladevorrichtung wie z.B. ein Tastenfeld oder eine Tastatur, die über eine optionale gesicherte physische Schnittstelle an die CF angeschlossen ist, installiert wird.
- Arbeitsschlüsselregister - Aus Gründen der Leistungsfähigkeit verfügt das System über Arbeitsregister, die jeweils 128 Bit groß sind und die Schlüssel enthält, mit denen gerade gearbeitet wird, damit schnell darauf zugegriffen werden kann. Zum Beispiel wird ein Schlüssel, der zur Chiffrierung von Daten benutzt wird, bei der erstmaligen Verwendung in chiffrierter Form in die CF geladen. Dann wird er dechiffriert, und der unverschlüsselte Wert kann in einem der Arbeitsschlüsselregister gespeichert werden. Wenn der Schlüssel erneut zum Chiffrieren oder Dechiffrieren der Daten benutzt wird, kann dieser unchiffrierte Schlüssel schnell aus einem speziellen Arbeitsschlüsselregister abgerufen werden, so daß eine neuerliche Dechiffrierung des Schlüssels vor der Verwendung überflüssig ist.

- Programm-MDC-Register (PMDC Reg) - Das Programm-MDC-Register umfaßt 64 Bit und enthält den MDC des Programms, das in den Programmspeicher in der CF geladen werden soll.
- Datei-MDC-Register (DMDC Reg) - Das Datei-MDC-Register umfaßt 64 Bit und enthält den MDC der Dateien, deren Integrität durch das CFAP geprüft wird. Dies sind in der Regel mindestens die Schlüsselspeicherdateien.
- # Chiffrieranweisungen und Algorithmen zur Prüfung von Steuervektoren - Der Anweisungssatz und die Algorithmen zur Prüfung von Steuervektoren sind in der sicheren Chiffrier-  
vorrichtung implementiert und im Chiffrieranweisungsspeicher 10, einem Direktzugriffsspeicher, gespeichert. Sie werden in einem Mikroprozessor wie z.B. einem Intel 80286 ausgeführt, der als Chiffrierverarbeitungseinheit 16 dienen kann. Die Steuervektor-Prüfeinheit 14 kann ebenfalls in der Chiffrierverarbeitungseinheit 16 oder in einem zweiten Mikroprozessor wie z.B. einem Intel 80286, der als Steuervektor-Prüfeinheit 14 fungiert, implementiert sein.
- # Programmspeicher und Verarbeitungsmaschine - Das System kann auch einen Speicher innerhalb der CF, in dem Programme des Benutzers gespeichert werden, sowie eine Verarbeitungsmaschine zur Ausführung dieser Programme enthalten. Ein Beispiel dafür ist ein Programm oder ein Makro zur Ausführung neuer Algorithmen zur Überprüfung der persönlichen Identifikationszahl oder für neue Identifikationszahlformate.
- # Zufallszahlengenerator 26 - Der Zufallszahlengenerator ist eine algorithmische Prozedur zur Erzeugung einer 64 Bit langen Pseudozufallszahl. Der Algorithmus selbst ist nicht geheim, verwendet aber zwei 128 Bit lange geheime Schlüssel und einen 64 Bit umfassenden nicht geheimen Inkrementalzähler. Obwohl also einige Komponenten nicht geheim sind, ist

die Integrität und die richtige Verwaltung des Zählers entscheidend für die Sicherheit.

Die Chiffriervorrichtung CF ist eine sichere Implementierung, die den Datenchiffrieralgorithmus enthält, und ein Speicher für eine geringe Anzahl von Schlüssel- und Datenparametern im Chiffrieranweisungsspeicher 10. Auf diesen kann nur über nicht manipulierbare Schnittstellen (geschützt vor unbefugtem Eindringen, Umgehung und Täuschung) zugegriffen werden, die die Eingabe von Verarbeitungsanforderungen, Schlüsseln und Datenparametern und den Empfang umgewandelter Ausgabedaten erlauben.

Der ANSI Datenchiffrieralgorithmus (DEA) ist ein Standardchiffrieralgorithmus für kommerzielle Datenschutzprodukte. Der DEA ist ein symmetrischer Blockchiffrieralgorithmus, der mit einem 56 Bit langen geheimen Schlüssel 64 Bit Klartext so chiffriert, daß der verschlüsselte Text ebenfalls 64 Bit lang ist. DEA-Schlüssel werden in der Regel mit einem Paritätsbit pro Byte gespeichert, so daß ein 64 Bit langer Schlüssel entsteht. Der DEA bildet die Grundlage für den vom National Bureau of Standards genehmigten bundesweiten Datenchiffrierstandard und wird auch als DES bezeichnet.

Die Chiffriervorrichtung muß dem Mißbrauchsversuch eines böswilligen Insiders mit beschränktem Zugriff zur Chiffrierungshardware standhalten können. "Beschränkt" bedeutet hier, daß die erlaubte Zugriffszeit nicht in Tagen oder Wochen, sondern in Minuten oder Stunden zu messen ist, und der Angriff kann nur an dem Ort erfolgen, an dem das System installiert ist, und nur unter Verwendung beschränkter elektronischer Ausrüstung. Ein Laborangriff an einem vom Gegner kontrollierten Ort unter Zuhilfenahme komplizierter elektronischer und mechnischer Geräte ist hier nicht gemeint.

Die Chiffriervorrichtung muß Versuche, sie physisch auszuspionieren oder in sie einzudringen, erkennen. Dafür stehen ver-

schiedene elektromechanische Erkennungsvorrichtungen zur Verfügung.

Die Chiffriervorrichtung muß dafür sorgen, daß alle intern gespeicherten unchiffrierten Schlüssel automatisch auf Null zurückgesetzt werden. Diese Rücksetzung muß automatisch erfolgen, sobald ein Spionage- oder Eindringungsversuch festgestellt wird. Außerdem muß die Chiffriervorrichtung so konstruiert sein, daß eine manuelle Rücksetzung der Schlüsselspeicher auf Null über die Schnittstelle an der Vorderseite möglich ist.

Die Chiffriervorrichtung enthält einen Hauptschlüssel KM. Alle anderen Schlüssel können auf einem Massenspeicher gespeichert werden, wobei sie unter einem Schlüssel chiffriert sind, der durch eine EXKLUSIV-ODER-Verknüpfung des Hauptschlüssels mit einem gültigen Steuervektor gebildet wird. Eine exemplarische Chiffriervorrichtung ist in der US-Patentschrift 4,386,234 mit dem Titel "Cryptographic Communications and File Security Using Germinals" von Ehrsam et al. beschrieben, die auf die IBM Corporation übertragen wurde und mittels Referenz Bestandteil der vorliegenden Patentschrift ist.

Das CFAP ist die Programmierschnittstelle zwischen der CF und dem Anwendungsprogramm. Da Benutzer keinen direkten Zugriff zu der Chiffriervorrichtung haben, ist das CFAP die Programmierschnittstelle, über die die Benutzer der CF Anweisungen zur Ausführung bestimmter Operationen erteilen können.

Dem CFAP ist der Schlüsselspeicher außerhalb der CF zugeordnet, in dem die chiffrierten Schlüssel gespeichert sind. Schlüssel im Klartext werden nicht außerhalb der CF gespeichert. Der Schlüsselspeicher 22 wird in der vorliegenden Patentschrift auch als "Chiffrierschlüsseldatei" (CKDS) bezeichnet.

# Das CFAP besteht aus folgenden Komponenten:

- Dem Schlüsselspeicherverwalter zur Verwaltung der im oben genannten Schlüsselspeicher gespeicherten Schlüssel.
- CFAP-Makros, durch die die Benutzer auf die CF zugreifen, um Chiffrierfunktionen auszuführen.

Zu den Anwendungsprogrammen zählen Anwendungsprogramme der Benutzer, einige Dienstprogramme wie z.B. ein Schlüsselinstallationsprogramm, und Datenübertragungsprogramme wie z.B. IBM VTAM.

Anwendungsprogramme der Benutzer bestehen aus Anweisungen, die bestimmte CFAP-Makros aufrufen, um eine bestimmte Aufgabe durchzuführen. Wie bereits erwähnt ist das CFAP die einzige Schnittstelle, durch die Anwendungsprogramme die Ausführung einer Chiffrierfunktion von der CF anfordern können. Ein Benutzer will beispielsweise eine Datei chiffrieren, bevor er sie an einen anderen Netzknoten sendet. Sein Anwendungsprogramm kann dafür eine Anweisung enthalten, die ein CFAP-Makro zur Generierung eines Schlüssels aufruft, und eine weitere Anweisung, die ein anderes CFAP-Makro zur Chiffrierung der Dateidaten unter dem generierten Schlüssel aufruft.

Ein anderes Beispiel für ein Benutzerprogramm ist ein Programm, das ähnlich wie das oben genannte Installationsprogramm die manuelle Installation von Schlüsseln auf dem System ermöglicht.

In Fig. 3 ist die Verwendung von Steuervektoren für die Schlüsselverteilung unter mehreren Anwendungen in einer Netzwerkumgebung dargestellt. Es ist zu beachten, daß aus Gründen der Kompatibilität einige Schlüssel ohne Steuervektoren vergeben werden müssen.

# Notation - Im folgenden wird folgende Notation verwendet:

ECB	Elektronisches Codebuch
-----	-------------------------



CBC	Chiffreblockkettung
KM	128-Bit-Hauptschlüssel
KEK	128-Bit-Schlüsselchiffrierschlüssel
K	64-Bit-Schlüssel
*K	128-Bit-Schlüssel
(*)K	64- oder 128-Bit-Schlüssel
KD	64-Bit-Datenchiffrierschlüssel
KK	64-Bit-Schlüsselchiffrierschlüssel
*KK	128-Bit-Schlüsselchiffrierschlüssel
KKo	64-Bit-Schlüsselchiffrierschlüssel mit Off- set
*KKo	128-Bit-Schlüsselchiffrierschlüssel mit Off- set
(*)KKo	64- oder 128-Bit-Schlüsselchiffrierschlüssel mit Offset
*KKNI	partieller beglaubigender 128-Bit-Schlüssel- chiffrierschlüssel
*KN	128-Bit-Beglaubigungsschlüssel, gleichbedeu- tend mit *KKNIo
Cx	64-Bit-Steuervektor
CxL	Linker 64-Bit-Steuervektor
CxR	Rechter 64-Bit-Steuervektor
XOR oder xor	EXKLUSIV-ODER-Verknüpfung
or	Logische ODER-Verknüpfung
X '0'	Hexadezimalschreibweise
11	Verkettungsoperation
[x]	Optionalen Parameter x
not =	Ungleich
E oder e	Einfache Chiffrierung
D oder d	Einfache Dechiffrierung
EDE oder ede	Dreifach-Chiffrierung
DED oder ded	Dreifach-Dechiffrierung

## # Gleichungen

Die Funktion aller Anweisungen ist in folgender Form mathematisch dargestellt:

$$I_1, I_2, I_3, I_4, \dots \rightarrow O_1, O_2, O_3, \dots$$

wobei  $I_1, I_2, I_3, \dots$  die Eingabedaten für die Funktion und  $O_1, O_2, O_3, \dots$  die Ausgabewerte der Funktion sind.

$KM.Cx$        $(KML \text{ XOR } Cx) \parallel (KMR \text{ XOR } Cx) = KMY \parallel KMX$   
wobei       $KML$  = linke 64 Bit des Hauptschlüssels  $KM$ ,  
             $KMR$  = rechte 64 Bit des Hauptschlüssels  $KM$ ,  
             $KMY$  =  $KML \text{ XOR } Cx$   
             $KMX$  =  $KMR \text{ XOR } Cx$   
            ist.

$e*KM.Cx(\text{Schlüssel})$   
 $e*KM.Cx(\text{Schlüssel}) = eKMY(dKMX(eKMY(\text{Schlüssel})))$   
wobei       $KMY$  =  $KML \text{ XOR } Cx$   
             $KMX$  =  $KMR \text{ XOR } Cx$   
            Schlüssel = 64-Bit-Schlüssel ist.

$e*KEKn.CX(\text{Schlüssel})$   
 $e*KEKn.Cx(\text{Schlüssel}) = eKEY(dKEKX(eKEY(\text{Schlüssel})))$   
wobei       $KEY$  =  $KEKnL \text{ XOR } CxL$   
             $KEKX$  =  $KEKnR \text{ XOR } CxR$   
            Schlüssel = 64-Bit-Schlüssel ist.

$e*KM.CxL(KEKnL)$   
 $e*KM.CxL(KEKL) = eKMY(dKMX(eKMY(KEKnL)))$   
wobei       $KEKL$  = linke 64 Bit des  $KEK$   
             $KMY$  =  $KML \text{ XOR } CxL$   
             $KMX$  =  $KMR \text{ XOR } CxL$  ist.

$e*KM.CxR(KEKnR)$

```

e*KM.CxR(KEKR) = eKMY(dKMX(eKMY(KEKnR)))
wobei      KEKR = rechte 64 Bit des KEK
           KMY = KML XOR CxL
           KMX = KMR XOR CxL ist

```

```

e*KEKo(Schlüssel)

```

```

e*KEKo(Schlüssel) = eKEKLo(dKEKRo(eKEKLo(Schlüssel))
wobei      KEKLo = KEKL XOR cntr
           KEKRo = KEKR XOR cntr
           cntr = impliziter 64-Bit-Schlüsselmel-
                 dungs-zähler für KEK
           Schlüssel = 64-Bit-Schlüssel ist.

```

#### # Kryptographische Schlüsseltrennung

Die Schlüssel sind kryptographisch nach Schlüsselart und Verwendungsattributen getrennt.

1. Die Architektur gewährleistet, daß Chiffrierschlüssel nur auf die vorgeschriebene und beabsichtigte Weise benutzt werden können.
2. Interne Trennung im Gegensatz zur Trennung bei der Übertragung. Intern (d.h. innerhalb der Chiffriervorrichtung) sind die Schlüssel durch Steuervektoren oder andere geeignete/-gleichwertige Mechanismen getrennt. Bei der Übertragung sind die Schlüssel mittels Steuervektoren getrennt.
3. Erzwingung mittels Hardware im Gegensatz zur Erzwingung mittels Software. Eine gewisse kryptographische Trennung ist hardwaremäßig implementiert; eine weitere kryptographische Trennung kann durch Software implementiert werden.
4. Steuervektor-Schlüsselarten und Kompatibilitätsmodus-Schlüsselarten. Damit die Kompatibilitätsmodus-Schlüssel-

arten die Sicherheit nicht beeinträchtigen, werden bestimmte Regeln bezüglich der Generierung, Verteilung und Verwendung dieser Schlüsselartenklassen festgeschrieben.

5. Gemäß der Erfindung sind folgende Schlüsseltrennungen erforderlich:

- a) Daten/Vertraulichkeit. ENCIPHER von DECIPHER, erlaubt Protokolle mit öffentlichem Schlüssel wie z.B. Mailbox, elektronische Abstimmungen und Weiterleitung.
- b) Daten/MAC. MACGEN von MACVER, berücksichtigt die Nicht-Zurückweisung (entspricht elektronischer Unterschrift).
- c) Daten/XLT. Erlaubt die Einrichtung eines sicheren Umwandlungskanals, in dem zwischengeschaltete Geräte die chiffrierten Daten nicht dechiffrieren können.
- d) Daten/Kompatibilität. Läßt den Kompatibilitätsmodus zu, ohne die Sicherheit anderer Datenschlüssel zu beeinträchtigen.
- e) Daten/ANSI. Erlaubt die Koexistenz der Schlüsselverwaltung nach ANSI X9.17 und der nicht dem ANSI X9.17-Protokoll entsprechenden Schlüsselverwaltung ohne Sicherheitsverlust auf beiden Seiten.
- f) Schlüsselchiffrierschlüssel. KEK-Sender von KEK-Empfänger.
- g) PIN-Schlüssel. PIN-Generierungsschlüssel vom PIN-Chiffrierschlüssel.

Für Fig. 4 bis Fig. 9 wird folgende Notation verwendet:

# Notation:

Bei jeder Linie, die aus einem Kästchen kommt, steht ein Trennungsbuchstabe und eine Prioritätszahl. Der Trennungsbuchstabe entspricht der nachstehenden Beschreibung. Der Prioritätszahlenbereich (1 bis 4) ist folgendermaßen zu interpretieren:

1. Absolut notwendig
2. Dringend empfohlen
3. Empfohlen
4. Wünschenswert

#### # Grundsätzliche Schlüsseltrennung

In Fig. 4 ist die grundsätzliche Trennung der Chiffrierschlüsselarten dargestellt. Dabei gilt folgendes:

1. A. Datenschlüssel : KEKs und PIN-Schlüssel - Wenn Datenschlüssel (KD) nicht von KEKs und PIN-Schlüsseln getrennt sind, kann die Daten-Dechiffrierfunktion in Verbindung mit Datenschlüssel zur Dechiffrierung von KEKs und PINs mißbraucht werden.
2. B. Schlüsselchiffrierschlüssel : PIN-Schlüssel - Wenn KEKs (Schlüsselchiffrierschlüssel) nicht von den PIN-Schlüsseln getrennt sind, kann ein Außenstehender einen chiffrierten PIN-Block abfangen und ihn anstelle eines chiffrierten KD dechiffrieren. Ein Insider könnte in der Chiffrierschlüsseldatei des Empfängerkontakts den chiffrierten gespeicherten KEK zukünftig durch den chiffrierten gespeicherten PIN-Schlüssel ersetzen. Der PIN-Block würde dann im Empfängerkontakt wiederhergestellt und als Datenschlüssel verwendet. Danach wären die Daten, die mit diesem PIN-Block als Datenschlüssel chiffriert werden, wesentlich leichter einem Schlüsselerschöpfungsangriff auszusetzen, da die Variabilität von PINs (in der Regel vier bis sechs Dezimalstellen) wesentlich geringer ist als die eines zufälligen 56-Bit-Datenschlüssels.

#### # Datenschlüsseltrennung

Fig. 5 ist das Flußdiagramm der Datenschlüsseltrennung. Für die Trennung sprechen folgende Gründe:

1. A - Identifikationsüberprüfung : Vertraulichkeit - Ein Insider, der unverschlüsselte Daten und den entsprechenden mit einem MAC-Schlüssel chiffrierten Text in Erfahrung bringen kann, kann einen Angriff gegen die MAC-Prozedur verüben. Es wäre beispielsweise möglich, betrügerische Nachrichten und MACs, die durch den MAC-Algorithmus als richtig identifiziert und akzeptiert werden, zu entwerfen. Die zur Chiffrierung bzw. Dechiffrierung von Daten benutzten Datenschlüssel dürfen deshalb nicht zur Identifikationsüberprüfung von Daten eingesetzt werden. Wenn bei der Übertragung ein MAC-Schlüssel durch einen abgefangenen Datenschlüssel ersetzt werden kann, könnte der übermittelte (unter diesem Datenschlüssel) chiffrierte Text zur Erzeugung einer betrügerischen Nachricht und eines falschen MAC mißbraucht werden.

B - XLT Chiffretext : Vertraulichkeit - XLT Chiffretext impliziert per definitionem die Verwendung eines Datenschlüsselpaares KD1 und KD2, wobei der unter KD1 verschlüsselte Chiffretext unter KD1 dechiffriert und anschließend unter KD2 erneut chiffriert wird, ohne daß die Daten dem aufrufenden Anwendungsprogramm zugänglich gemacht werden. Andernfalls könnte XLT Chiffretext mit den vorhandenen Chiffrier- und Dechiffrierfunktionen ausgeführt werden.

C - ANSI : alle übrigen - ANSI-Schlüssel haben ein eigenes Schlüsselverteilungsprotokoll und zusätzlich einen möglichen Verwendungszweck, die als ANSI COMBINE KEYS bezeichnet werden. Diese Unterschiede erfordern einen separaten Pool für alle ANSI-Schlüssel.

D - Datenkompatibilität : alle übrigen - Datenkompatibilitätsschlüssel existieren, weil die Kompatibilität mit älteren Systemen wie IBM CUSP/3848, IBM PCF und IBM 4700 notwendig ist. Da die erzwungene interne Trennung bei diesen Systemen sich nicht auf die Ebene der Unterscheidung zwischen MAC-Schlüsseln und Vertraulichkeitsschlüsseln erstreckt, müssen diese Schlüssel von den CV-Schlüsseln, die eine solche höhere Trennungsebene unterstützen, abgegrenzt werden.

2. B - MACGEN : MACVER - Liefert ein Prüfprotokoll als "Beleg", wer eine Nachricht und MAC gesendet hat (Nicht-Zurückweisung). Dieses Verfahren ist nicht sicherer als die CF und geht von einem gegenseitigen Vertrauen in die Integrität und Geheimhaltung der in der CF gespeicherten Schlüssel aus.
3. C - Chiffrierung : Dechiffrierung - Chiffrier- und Dechiffrierfunktion sind tatsächlich getrennt, so daß Daten unter einem Datenschlüssel chiffriert werden können, ohne daß der gleiche Datenschlüssel auch zur Dechiffrierung zugelassen ist. Ein Datenschlüssel, der nur zur Chiffrierung von Daten benutzt werden kann, könnte beispielsweise für ein Abstimmungsverfahren verwendet werden, in dem eine Stimme abgegeben und weitergereicht wird. Ein nur für die Dechiffrierung zugelassener Schlüssel könnte beispielsweise in einer Umgebung benutzt werden, in der ein Benutzer bestimmte Daten nur lesen, aber nicht schreiben darf.

#### # Trennung von PIN-Schlüsseln

Fig. 6 ist das Flußdiagramm der PIN-Schlüsseltrennung. Für die Trennung gibt es folgende Gründe:

1. A - PIN-Generierungsschlüssel : PIN-Chiffrierschlüssel - Ein Insider, der veranlassen könnte, daß einem PIN-Block ein gültiger ID-Wert zugewiesen und dieser PIN-Block dann unter einem PIN-Generierungsschlüssel anstelle eines PIN-Chiffrierschlüssels chiffriert wird, könnte PINs offenlegen.
2. B - PIN-Generierungsfunktion : PIN-Chiffrierfunktion - Bei der PIN-Generierung ermöglicht das Attribut "PIN chiffrieren" die Trennung von PIN-Generierungsschlüsseln, mit denen PINs im Klartext erzeugt werden, von PIN-Generierungsschlüsseln, die in jedem Fall chiffrierte PINs erzeugen müssen.
3. C - PIN-Block erstellen und PIN generieren : PIN neu formatieren, PIN überprüfen und PIN umwandeln - Diese Unterscheidung ermöglicht die Verwendung von PIN-Chiffrierschlüsseln in Verbindung mit routinemäßigen PIN-Verarbeitungsfunktionen wie Neuformatierung, Überprüfung und Umwandlung von PINs, ohne daß PIN-Schlüssel verwendet werden dürfen, oder die Erstellung oder anderweitige "Einführung" von PINs in das Netzwerk in einer elektronischen Geschwindigkeit. Auf diese Weise kann die Herstellung von Übersetzungslisten von Klartext-PINs und chiffrierten PINs bei elektronischer Geschwindigkeit verhindert werden. Diese wären beim Ausspionieren und Wiederherstellen von PINs ohne direkte Dechiffrierung nützlich. Es muß streng kontrolliert werden, wo, wann und unter welchen Bedingungen PINs in das System eingeführt werden können.
4. D - PIN-Block erstellen : PIN generieren - Die Einführung von PINs in das Netzwerk kann besser kontrolliert werden. Ein Knoten, an dem PIN-Blöcke erstellt werden müssen, hat nicht unbedingt die Berechtigung oder die Notwendigkeit, PINs zu generieren.



5. E - PIN neu formatieren und überprüfen : PIN umwandeln - Die PIN-Verarbeitungsfunktionen im Netzwerk können besser kontrolliert werden. Ein Knoten, an dem PINs umgewandelt werden müssen und dürfen, hat nicht unbedingt auch die Berechtigung, eine PIN neu zu formatieren oder zu überprüfen. Die letzteren beiden Funktionen können gemeinsam benutzt werden, um die Gültigkeit von PINs mittels eines internen Angriffs auslaufen zu lassen, wohingegen die Funktion zur Umwandlung von PINs von manchen Knoten verwendet werden kann, ohne die vollständigen Verarbeitungsmöglichkeiten aus der Hand zu geben.

#### # Trennung von Schlüsselchiffrierschlüsseln

Fig. 7 ist das Flußdiagramm der Trennung von Schlüsselchiffrierschlüsseln. Für diese Trennung gibt es folgende Gründe:

1. A - Beglaubigung - Nichtbeglaubigung - Ein Insider könnte veranlassen, daß ein Schlüssel, der für die Verwendung mit einem Offset bestimmt ist, auch ohne Offset/Beglaubigung benutzt werden kann, so daß die Variante des Schlüsselwertes gleich einem alten Offset-Zählerwert ist. Umgekehrt könnte ein Insider veranlassen, daß ein Schlüssel, der nur für die Verwendung ohne Offset bzw. ohne Berechtigung bestimmt ist, in einem Offset-Prozeß verwendet wird, so daß der Offset des Schlüssels gleich einer Variante ist, die nicht über einen privilegierten Modus in der Chiffriervorrichtung oder durch einen Eintrag in der autorisierten Variantentabelle erstellt oder generiert werden sollte.
2. B - CV-KEKS : nicht-CV-KEKS - Von einer CV-Chiffriervorrichtung ausgeführte Chiffrieroperationen zur Unterstützung anderer Nicht-CV-Netzwerkknoten müssen nicht unbedingt eine Schwachstelle für die Sicherheit des CV-Netzwerkknotens

sein. Ein CV-System muß beispielsweise den Import von Vertraulichkeits- und Identifikationsüberprüfungsschlüsseln (MAC-Schlüsseln) von einem Nicht-CV System des Typs IBM 4700 oder IBM 3848 unterstützen. In allen Fällen werden diese Datenschlüssel unter der Variante 0 des gemeinsamen Mehrdomänenschlüssels empfangen. Sind die Mehrdomänenschlüssel dieser Nicht-CV-Systeme nicht kryptographisch von den Mehrdomänenschlüsseln von CV-Systemen getrennt, wäre es möglich, einen für einen bestimmten Verwendungszweck (festgelegt durch Variante 0 des Mehrdomänenschlüssels) vorgesehenen Datenschlüssel für ein CV-System zu importieren und ihn für einen anderen Zweck (angegeben durch andere Varianten) zu benutzen.

3. C - KEK Sender : KEK Empfänger - Es wird die gleiche Richtungsgebundenheit für Mehrdomänenschlüssel beibehalten wie beim aktuellen IBM 3848/CUSP und IBM PCF. Außerdem ist eine bessere Kontrolle zur Verhinderung der unbefugten Erstellung von KEKs für beide Übertragungsrichtungen möglich.
4. D - GKS/XLATE : RFMK/GKS/XLT - Diese Trennung erlaubt KEKs zur Unterstützung von GKS und XLT zur Auslieferung von Datenschlüsseln, ohne daß zugleich der Export vorhandener unter dem Hauptschlüssel (oder einer Variante des Hauptschlüssels) gespeicherter Datenschlüssel erlaubt sein muß. Die vom CV-System benutzten Datenschlüssel werden somit nicht dem Export ausgesetzt, sofern dies nicht erforderlich oder gewünscht ist.
5. E - GKS (nur Export)/XLT : GKS (allgemeine Verwendung) - Diese Trennung erlaubt es einem Knoten, als Schlüsselveilungszentrale (KDC) oder Schlüsselumwandlungszentrale (KTC) zu fungieren, ohne daß er die Möglichkeit hat, die generierten Datenschlüssel innerhalb des generierenden Knotens zu benutzen.

6. F - RTMK : RFMK bei IBM 3838/CUSP (und kompatiblen Systemen) - Diese Trennung unterstützt die Richtungsgebundenheit bei IBM 3848/CUSP und anderen Systemen, die mit CV-Systemen kompatibel sind.

Fig. 8 ist eine Übersicht über die Schlüsseltrennungen und ihrer relativen Prioritäten. Die höchste Priorität ist '1', die niedrigste '4'.

Fig. 9 ist eine Zusammenfassung der Flußdiagramme zur Schlüsseltrennung. Die "Blätter" des Baums bezeichnen die Chiffrieranweisungen, die von den verschiedenen Schlüsselarten Gebrauch machen.

### Steuervektoren

#### # Prinzip der Steuervektoren

Der Steuervektor ist eine 64 Bit lange, nicht geheime Chiffriervariable zur Kontrolle der Schlüsselverwendung. Jedem im Chiffriersystem definierten Schlüssel K ist ein Steuervektor C zugeordnet, d.h. Schlüssel und Steuervektor definieren ein Tupel (K, C).

Jeder Steuervektor legt eine Steuervektorart CV TYPE, die grob definiert, wie der Schlüssel verwendet werden darf, sowie die Regeln, wie der betreffende Schlüssel an ein anderes System übermittelt werden darf, fest. Ein Schlüssel kann als Datenschlüssel, als Schlüsselchiffrierschlüssel des Senders, als Schlüsselchiffrierschlüssel des Empfängers, als PIN-Chiffrierschlüssel, als Zwischen-ICV, als Schlüsselteil oder als Token fungieren. Zusätzliche Bits im Steuervektor definieren exakt, in welchen Chiffrieranweisungen und Parametereingaben der Schlüssel verwendet werden kann. Andere Bits regeln den Export des Schlüssels, d.h., ob der Schlüssel exportiert werden darf oder nicht.

Der Steuervektor ist über eine spezielle Chiffrierfunktion kryptographisch mit dem Schlüssel gekoppelt. Angenommen, der Schlüssel  $K$  ist in einem Schlüsselspeicher gespeichert, dann ist  $K$  unter einem durch EXKLUSIV-ODER-Verknüpfung des Steuervektors mit dem Hauptschlüssel gebildeten Schlüssel chiffriert, d.h.,  $K$  ist als Tupel  $(eKM.C(K), C)$  gespeichert; dabei bedeutet  $KM.C$  soviel wie  $KM \text{ xor } C$ . Bei der Übertragung von  $K$  von einem Gerät an ein anderes wird eine ähnliche chiffrierte Form verwendet. In diesem Fall wird der Hauptschlüssel  $KM$  durch einen Schlüsselchiffrierschlüssel  $KEK$  ersetzt, wobei  $KEK$  ein Schlüssel ist, den sowohl der Sender als auch der Empfänger besitzt.  $K$  wird also als Tupel  $(eKEK.C(K), C)$  übertragen. Die Architektur verlangt nicht, daß der Steuervektor mit dem Schlüssel zusammen gespeichert oder übertragen wird, wenn sein Wert implizit aus dem Kontext definiert ist oder mittels verfügbarer schlüsselbezogener Informationen abgeleitet werden kann.

Da der Steuervektor  $(C)$  über die chiffrierte Form  $eKM.C(K)$  oder  $eKEK.C(K)$  eng mit dem Schlüssel  $(K)$  gekoppelt ist, kann  $K$  offensichtlich nicht aus der chiffrierten Form rekonstruiert werden, sofern  $C$  nicht richtig angegeben ist. Wenn also das Tupel  $(EKM.C(K), C)$  als Eingabe an eine angeforderte Chiffrieranweisung übergeben wird, prüft die Chiffriervorrichtung zuerst den für  $C$  angegebenen Wert und stellt fest, ob die angeforderte Verwendung des Schlüssels zulässig ist. Nur wenn dies der Fall ist, wird  $C$  zum Dechiffrieren von  $eKM.C(K)$  in den Klartextwert von  $K$  innerhalb der Chiffriervorrichtung verwendet. Wird für  $C^*$  ein falscher Wert angegeben, kann die Chiffriervorrichtung zwar kurzfristig dazu überlistet werden,  $C^*$  zu akzeptieren, aber  $K$  wird dann nicht richtig rekonstruiert. Ein Benutzer hat also keine Möglichkeit, den richtigen Wert von  $K$  wiederherzustellen, wenn er nicht den richtigen Wert für  $C$  angibt. Das Chiffrierprinzip ist also die Grundlage, auf der die gesamte Architektur aufbaut; zusätzliche Sicherheit kommt nach Bedarf hinzu.

Der Steuervektor ist eine kompakte Datenstruktur zur Festlegung der Verwendungsattribute eines Chiffrierschlüssels. Der Steuervektor ist über einen Verschlüsselungsprozeß kryptographisch an den Schlüssel gekoppelt. Dieser Prozeß ist so aufgebaut, daß der Schlüssel nur dann richtig dechiffriert werden kann, wenn der Steuervektor richtig angegeben wird. (Die Veränderung eines einzigen Bits im Steuervektor hat die Rekonstruktion eines ganz anderen Schlüssels zur Folge.)

#### # Steuervektorprüfung

Der Steuervektor ist so aufgebaut, daß nur ein minimaler Prüfungsaufwand erforderlich ist. Die Verwendungszweckbits sind so definiert und strukturiert, daß jedes Verwendungsattribut von sich aus einen bestimmten Verwendungszweck erlaubt oder verweigert. Die Fähigkeit zur Chiffrierung von Daten über die Anweisung "Daten chiffrieren" wird durch ein einziges "Chiffrier"-Bit unter der Kontrolle des Steuervektors der Art/Unterart "Daten/-Vertraulichkeit" gesteuert.

Jedes Verwendungsattribut wird also unabhängig von allen anderen Verwendungsattributen definiert. Dadurch ist eine Steuervektorprüfung gewährleistet, bei der jede Anweisung nur die von der angeforderten Funktion benötigten Verwendungsattribute prüft.

Ein Verfahren, bei dem Verwendungsattribute nur aktiviert werden, wenn bestimmte andere Attribute aktiv bzw. inaktiv sind, wird ausdrücklich vermieden, da dies den Prüfungsaufwand erhöhen würde. Die Querkontrolle von Attributen zweier oder mehrerer Steuervektoren ist zwar manchmal erforderlich, kann aber auf ein Minimum reduziert werden.

Um die Steuervektorprüfung zu erleichtern und zu vereinfachen, wird jeder Chiffrieranweisung bei Bedarf ein "Modus"-Parameter übergeben, in dem ein angegebener Verwendungszweck des Schlüssels bzw. der Schlüssel, die als Parameter an die Anweisung übergeben werden, deklariert wird. So wird jeder Steuervektor

gemäß dem angegebenen "Modus" geprüft. Dadurch wird die aufwendige Querkontrolle mehrerer Steuervektorattribute zur Gewährleistung der Konsistenz überflüssig.

Der Aufbau der Steuervektoren arbeitet nach einem Prinzip, daß keine Chiffrieranweisung einen Steuervektor generieren kann. Alle Steuervektoren werden den Chiffrieranweisungen als Parametereingaben präsentiert.

Wenn möglich befinden sich gleiche Verwendungsattribute und Felddefinitionen unabhängig von der Art des Steuervektors immer an der gleichen Bitposition des Steuervektors. Die Anweisung zur Umwandlung von Chiffretext fragt beispielsweise beim Daten/Vertraulichkeits-CV und beim Daten/Umwandlungs-CV die gleichen Bitpositionen ab, obwohl die Verwendungszweckbits für den Daten/Vertraulichkeits-CV "E" und "D" lauten, während sie für den Daten/Umwandlungs-CV "XOUT" und "XIN" lauten.

- # CV-Struktur - Im allgemeinen wurde die Struktur der Steuervektoren (einschließlich der Formate, Felder und Bitzuordnungen) so definiert, daß die Steuervektorprüfung minimiert und erleichtert wird und gleichzeitig ein hoher Sicherheitsstandard erreicht wird. Die CV-Struktur ist sozusagen mit dem höchsten Freiheitsgrad im Entwicklungsprozeß variabel.

Im Steuervektor sind folgende Strukturmerkmale verwirklicht:

1. Vertikale Trennung - Der Steuervektor besitzt ein "CV-Art"-Feld, in dem eine vertikale Trennung innerhalb der Steuervektorstruktur vorgenommen wird, ähnlich wie bei der Trennung durch Varianten. Steuervektorarten sind auf intuitiven Linien definiert, die der vorhandenen Schlüsselterminologie und der Schlüsselverwaltung folgen. Die vertikale Trennung wird unter der CA jedoch nur bei Bedarf implementiert, um so die Architektur einfach und die Regeln für die Steuer-

vektorprüfung übersichtlich zu gestalten. Indem zuerst grobe Klassen von CV-Hauptarten (z.B. Datenschlüssel, Schlüsselchiffrierschlüssel, PIN-Schlüssel) und dann erst Unterarten und Verwendungsattribute innerhalb der CV-Art definiert werden, können die Regeln für die Steuervektorprüfung in ähnlicher Weise optimiert werden, wie das Prinzip "Teilen und Herrschen" oft effektiver ist als rohe Gewalt.

2. Horizontale Trennung - Der Steuervektor eignet sich ideal als Datenstruktur für die Angabe der Verwendungsattribute, die für einen Schlüssel (oder eine andere kryptographische Variable) gelten sollen. Innerhalb der CA geschieht dies, indem für jede Chiffrieranweisung, für die der Schlüssel als Eingabe verwendet werden kann (oder für jeden Schlüsselparameter in der Anweisung, falls mehrere Schlüsselparameter beteiligt sind) ein Bit im Steuervektor gesetzt wird. Der Bitwert "1" bedeutet, daß eine bestimmte Verwendung des Schlüssels durch die CF "aktiviert" ist; der Bitwert "0" hingegen bedeutet, daß die Verwendung des Schlüssels durch die CF "deaktiviert" ist. Diese Form der Strukturierung von Steuervektoren wird als horizontale Trennung bezeichnet.
3. Codierte Felder - Ein Feld, das aus zwei oder mehr Bits besteht, ist manchmal aus Sicherheitsgründen codiert. Ein codiertes Feld hat die Eigenschaft, daß die einzelnen Bits selber keine eigene Bedeutung besitzen, daß sie aber zusammen einen Satz möglicher Werte definieren. Codierte Felder haben den Vorteil, daß sie einander ausschließende Ereignisse definieren, da das Feld immer nur einen einzigen Wert enthalten kann. Codierte Felder haben aber auch den potentiellen Nachteil, daß die Steuervektorprüfung aus leistungsmäßiger Sicht nicht immer optimiert ist. Manchmal sind codierte Felder jedoch notwendig, damit gewährleistet ist, daß Verwendungsattribute nicht zu unpassenden Kombinationen zusammengefügt werden können, die einem Angriff auf

das Chiffriersystem den Weg bereiten oder das System schwächen können.

4. Schutz vor nicht vom System generierten Schlüsseln - Das Verfahren der Kopplung von Steuervektor und Schlüssel ist so aufgebaut, daß die Steuervektorprüfung nicht zwischen einem vom System generierten Schlüssel (durch KGEN oder GKS) und einem nicht vom System generierten Schlüssel zu unterscheiden vermag. Es gibt deshalb in dieser Architektur eine "Hintertür" zur Generierung von Schlüsseln und Steuervektoren. Sie besteht darin, einen "ausgewählten" Steuervektor und eine Zufallszahl zu definieren, die dann als ein Schlüssel dargestellt wird, der mittels des Steuervektors auf die unter der Architektur beschriebenen Weise codiert ist. (Mit diesem Verfahren kann jedoch der tatsächlich innerhalb der CF rekonstruierte Schlüssel nicht kontrolliert werden.) Die sogenannte "Hintertür" der Schlüsselgenerierung ist vor allem lästig, auch wenn in einigen Fällen Angriffe auf die Chiffrierung denkbar sind, wenn in der Architektur keine zusätzlichen Schutzmaßnahmen getroffen werden. Es wäre ein leichtes, eine Architektur zu definieren, in der diese "Hintertür" (ein für allemal) aus der Welt geschafft ist; dies hätte jedoch eine zusätzliche Komplexität mit möglichen Sicherheitslücken und einen zusätzlichen Verarbeitungsaufwand zur Folge. Die CA verfolgt deshalb einen eher praktischen Ansatz, indem nämlich die Schlüsselgenerierung über die "Hintertür" nur dort verhindert wird, wo dies aus Sicherheitsgründen erforderlich ist. Auf diese Weise wird ein ausgewogenes Verhältnis zwischen Sicherheit, Komplexität und Leistung erzielt. Verfahren zur Vermeidung von kryptographischen Schwächen, die durch die Schlüsselgenerierung durch die "Hintertür" eingeführt werden, sind:

- a) Wenn nötig werden nicht miteinander verträgliche Verwendungsattribute in einem einzigen Steuervektor auf zwei



Steuervektoren aufgeteilt. Die Anweisung GKS beinhaltet eine Prüfung, die verhindert, daß sogenannte schlechte Schlüsselpaarkombinationen generiert werden.

- b) Bei Bedarf werden nicht miteinander verträgliche Verwendungsattribute in einem einzigen Steuervektor zu einem einzigen codierten Feld zusammengefaßt.
  - c) Als letztes Mittel wird zusätzliche Redundanz eingeführt, um der CF die Gültigkeitsprüfung ihrer eigenen vom System generierten Schlüssel zu ermöglichen.
5. Gerade Parität für Steuervektoren - Für Steuervektoren wird gerade Parität erzwungen. Dadurch wird sichergestellt, daß die EXKLUSIV-ODER-Verknüpfung eines Schlüssels mit ungerader Parität und des Steuervektors einen internen Schlüssel mit ungerader Parität ergibt. Dies gewährleistet wiederum die Kompatibilität mit einer Hardware, die solche intern abgeleiteten Schlüssel auf ungerade Parität prüft (falls eine solche Prüfung erzwungen wird). Mit anderen Worten, die CA kann nicht garantieren, daß die Hardware keine ungerade Parität bei internen Schlüsseln erzwingt. Ein Steuervektor umfaßt 64 Bits, die von 0 bis 63 durchnummeriert sind. Als höchstwertiges Bit ist Bit 0 vereinbart. Acht der 64 Bits sind Paritätsbits.
6. Anti-Varianten-Bits - Dadurch wird die kryptographische Trennung zwischen Varianten und Steuervektoren gewährleistet, die bei einigen Implementierungen unvermeidlich in einem Knoten intern gemischt werden.
7. Vermeiden von Zielabbildungen - Der Steuervektoraufbau und die Verarbeitung des Steuervektors über den Chiffrieranweisungssatz vermeiden Fälle, bei denen CV-Felder mit mehreren Werten in einen einzigen Wert abgebildet werden. Einige Spezialfälle solcher Abbildungen sind erlaubt (z.B. bei den Anweisungen LCVA, RFMK und RTMK), wo die Sicherheit nicht gefährdet wird.

## # CFAP-Steuerung

Bestimmte Bits im Steuervektor sind für das CFAP reserviert. Diese Bits können vom CFAP für eine weitergehende Steuerung der Schlüsselverwaltung benutzt werden. Sie werden nicht von der CF geprüft, sondern allein vom CFAP verwaltet.

### Allgemeines Format für Steuervektoren

In Fig. 10 ist das allgemeine Format der Steuervektoren dargestellt. In der ersten Reihe der Tabelle sind die Felder aufgeführt, die bei den meisten Steuervektoren gleich sind. Sie werden im folgenden kurz beschrieben (Einzelheiten sind nachfolgenden Abschnitten zu entnehmen).

## # CV-Art

Dieses Feld bezeichnet die Art des Steuervektors und außerdem die Schlüsselart des Schlüssels, dem dieser Steuervektor zugeordnet ist. Das Feld "CV-Art" besteht aus Hauptart und Unterart.

Die Hauptarten von Steuervektoren sind:

- E
 - **Datenschlüssel** - Datenschlüssel werden zum Chiffrieren bzw. Dechiffrieren von Daten oder zur Identifikationsüberprüfung von Daten verwendet.
- **PIN-Schlüssel** - PIN-Schlüssel werden zum Chiffrieren oder Generieren von PINs verwendet.
- **Schlüsselchiffrierschlüssel** - Schlüsselchiffrierschlüssel werden zum Chiffrieren von Schlüsseln benutzt.
- **Schlüsselteil** - Ein Schlüsselteil ist ein Teil oder eine Komponente eines Schlüssels, der aber die gleiche Länge hat wie ein ganzer Schlüssel. Ein Schlüssel K kann beispielsweise zwei Schlüsselteile  $K_a$  und  $K_b$  besitzen, für die  $K_a \text{ XOR } K_b = K$  gilt.

- Zwischen-ICV - Ein Zwischen-ICV wird bei der MAC-Verarbeitung verwendet, um den Ausgabekettungszwischenwert OCV eines Segments oder einer Nachricht zu chiffrieren. Dieser OCV wird dann an das nächste Datensegment übergeben und als ICV verwendet. Dies geschieht, wenn eine Nachricht oder die Daten, mit denen ein MAC generiert oder überprüft werden soll, lang sind und in kürzere Segmente aufgegliedert werden müssen.
- Token - Tokens sind Variable zum Schutz der Integrität der in der Datenschlüsseldatei (einem Schlüsselspeicher für Datenschlüssel) gespeicherten Datenschlüssel. Sie tragen dazu bei, den Zugriff nicht dazu berechtigter Benutzer auf Datenschlüssel zu verhindern.

Die Unterart ist eine weitere Differenzierung der Schlüsselklassen der gleichen Hauptart. Ein Schlüssel der Hauptart "Datenschlüssel" kann zum Beispiel die Unterart "Vertraulichkeit" (für Chiffrierung und Dechiffrierung) oder die Unterart "MAC" (für die Datenidentitätsüberprüfung) oder die Unterart "XLATE Daten" (für die Umwandlung von Chiffretext) usw. haben. Wenn nicht zwischen Unterarten unterschieden wird, werden die Schlüssel gewöhnlich mit der Hauptart (z.B. Datenschlüssel, PIN-Schlüssel usw.) bezeichnet.

#### **# Exportkontrolle**

In diesem Feld wird angegeben, wie der Export des diesem Steuervektor zugeordneten Schlüssels geregelt wird und ob der Schlüssel überhaupt exportiert werden darf.

#### **# Von der CF erzwungene Verwendung**

In diesem Feld wird festgelegt, für welche CA-Funktionen der Schlüssel benutzt werden kann, und wie er verwendet werden darf. Ein Datenvertraulichkeitsschlüssel kann beispielsweise die Verwendungsattribute E = 1 und D = 1 besitzen, die besagen, daß der

Schlüssel in der Chiffrierfunktion und in der Dechiffrierfunktion zum Chiffrieren bzw. Dechiffrieren der Daten benutzt werden kann.

#### # AV (Anti-Variante)

In diesem Feld wird jeder gültige Steuervektor von den 64 vordefinierten Varianten unterschieden, die in Chiffriersystemen benutzt werden, die auf Variantenbasis arbeiten. Da bei jeder der 64 vordefinierten Varianten alle acht Bytes gleich sind, wird dadurch, daß der Wert in AV-Feld so gesetzt wird, daß mindestens zwei Byte des Steuervektors nicht gleich sind, ein gültiger Steuervektor von einer vordefinierten Variante unterschieden.

#### # Software-Bits

In diesem Feld werden die Steuervektorbits angegeben, die ausschließlich vom CFAP kontrolliert bzw. verwaltet werden. Das Softwarefeld wird nicht von der Hardware (CF) geprüft oder erzwungen. Wenn kein Steuervektor existiert, erstellt das CFAP aus der ihm (in der Regel über Parameter in einem Makro) übergebenen Information einen Steuervektor. Existiert bereits ein Steuervektor prüft das CFAP den Steuervektor (einschließlich des Softwarefeldes), um festzustellen, ob der Schlüssel auf die angegebene bzw. angeforderte Weise verwendet werden darf. Anders als die Software (CFAP) prüft die Hardware (CF) nur die Bits, die zu einer CA-Anweisung gehören; andere Verwendungszweckbits werden nicht geprüft.

#### # Länge

In diesem Feld wird festgelegt, ob es sich um einen 64-Bit-Steuervektor oder um einen erweiterten Steuervektor mit einer Länge von 128 Bit handelt. Bei der derzeitigen CA sind alle Steuervektoren 64 Bit lang. Dieses Feld wird jetzt definiert, damit der

Steuervektor in der Zukunft leichter vergrößert werden kann, wenn die derzeitige Länge von 64 Bit nicht mehr für die Definition des Steuervektors ausreicht.

#### # Reservierte Bits

Dieses Feld ist für eine zukünftige Verwendung durch das System reserviert.

#### # Paritätsvektor

Jedes Paritätsbit ist die gerade Parität der vorausgehenden 7 Bits des Byte.

Bei Steuervektoren für Schlüsselchiffrierschlüssel gibt es neben den oben genannten allen Steuervektoren gemeinsamen Feldern noch zwei weitere Felder, nämlich SCHLÜSSELFELD und VERBINDUNGSSTEUERUNG.

#### # Schlüsselfeld

In diesem Feld wird die Schlüssellänge (einfache oder doppelte Länge) festgelegt und angegeben, ob die dem Steuervektor zugeordnete Schlüsselhälfte die rechte oder die linke Hälfte des Schlüssels ist. Bei einem Schlüssel einfacher Länge ist die rechte Hälfte mit der linken und mit dem Schlüssel selbst identisch.

#### # Verbindungssteuerung

In diesem Feld wird angegeben, wie der diesem Steuervektor zugeordnete Schlüsselchiffrierschlüssel zur Übertragung anderer Schlüssel verwendet wird, und an welche bzw. von welcher Art von System (CV-System oder Nicht-CV-System) Schlüssel unter diesem Schlüsselchiffrierschlüssel gesendet oder empfangen werden können.

Die Beschreibungen in der zweiten und dritten Reihe der allgemeinen Tabelle und anderer in diesem Abschnitt beschriebener Tabellen sind nicht Bestandteil des Steuervektors. Sie sind hier aufgeführt, um folgende Informationen über die Felder des Steuervektors zu ergänzen:

- In der zweiten Reihe ist die Länge der Felder in Bit angegeben. Die Abkürzung 'b' steht für 'Bit'. 1b bedeutet 1 Bit, 3b bedeutet 3 Bit usw.
- In der dritten Reihe steht, ob das Feld von der Hardware (CF) oder von der Software (CFAP) geprüft wird.

#### Steuervektorformat für Datenschlüssel

Datenschlüssel sind in folgende Unterarten unterteilt:

- Datenkompatibilitätsschlüssel. Dies ist ein Datenschlüssel, der zur Wahrung der Kompatibilität mit älteren Systemen wie dem IBM 3838/CUSP oder dem IBM 4700 FCS benutzt wird. Da bei diesen Systemen keine kryptographische Trennung zwischen Vertraulichkeitsschlüsseln und Identifikationsüberprüfungsschlüsseln erfolgt, können mit diesem Schlüssel folgende Funktionen ausgeführt werden: Chiffrierung, Dechiffrierung, MAC-Generierung und MAC-Überprüfung. Dieser Steuervektor kann beim Export an andere Systeme (z.B. mittels der Anweisung RFMK) abgetrennt (d.h. bei der Übertragung durch CV = 0 ersetzt werden), während die Steuervektoren für alle anderen Datenschlüssel mit Ausnahme von ANSI-Datenschlüsseln nicht abgetrennt werden können.
- Vertraulichkeitsschlüssel. Mit diesem Schlüssel kann nur chiffriert und/oder dechiffriert werden.
- MAC-Schlüssel. Dieser Schlüssel wird nur zur Datenidentifikationsüberprüfung benutzt. Er kann also nur zur Generierung und/oder Überprüfung von MACs eingesetzt werden.

- Datenumwandlungsschlüssel (Daten-XLT-Schlüssel). Dieser Schlüssel dient zur Umwandlung von Chiffretext.
- ANSI-Schlüssel. Dieser Schlüssel wird in ANSI-Anwendungen verwendet. Er kann zum Chiffrieren und Dechiffrieren von Daten oder zur Generierung und Überprüfung von MACs benutzt werden. Er kann auch mit einem anderen ANSI-Schlüssel zu einem ANSI MAC-Schlüssel (d.h. einem ANSI-Datenschlüssel mit der Möglichkeit zur Generierung und Überprüfung von MACs) kombiniert werden. Dieser Steuervektor kann beim Export an andere Systeme durch die Anweisung ARFMK abgetrennt (d.h. bei der Übertragung durch CV = 0 ersetzt) werden, während die Steuervektoren für alle anderen Datenschlüssel mit Ausnahme von Kompatibilitätsschlüsseln nicht abgetrennt werden können.

Je nach CV-Unterart des Steuervektors haben die Bits im Feld VERWENDUNG eine spezielle Bedeutung, die im folgenden kurz erläutert wird.

#### Steuervektor für Vertraulichkeitsschlüssel

Das Format der Steuervektoren für Vertraulichkeitsschlüssel ist in Fig. 11 dargestellt. Im folgenden werden die einzelnen Felder und Teilfelder dieser Tabelle ausführlich beschrieben.

- # CV-Art - für Vertraulichkeitsschlüssel (Hauptart = "Datenschlüssel", Unterart = "Vertraulichkeit").
- # Exportkontrolle (regelt den Export dieses Schlüssels): Dieses Feld belegt ein Bit:
  - EXPORTKONTROLLE = 1: Dieser Schlüssel kann durch RFMK exportiert werden. Außerdem können die Anweisungen RFMK, RTMK und LCVA dieses Bit auf "0" setzen.

- EXPORTKONTROLLE = 0: Dieser Schlüssel kann nicht durch RFMK exportiert werden. Dieses Bit kann auch durch keine Anweisung in "1" geändert werden.

Als Beispiel soll angenommen werden, ein Knoten X generiert einen Schlüssel K und einen Steuervektor C und sendet diese an den Knoten Y.

#### # Verwendung

##### \* E

- E = 1: Dieser Schlüssel kann in der Anweisung ENCIPHER zum Chiffrieren von Daten benutzt werden.
- E = 0: Dieser Schlüssel kann nicht in der Anweisung ENCIPHER zum Chiffrieren von Daten benutzt werden.

##### \* D

- D = 1: Dieser Schlüssel kann in der Anweisung DECIPHER zum Dechiffrieren von Daten benutzt werden.
- D = 0: Dieser Schlüssel kann nicht in der Anweisung DECIPHER zum Dechiffrieren von Daten benutzt werden.

#### # AV (Anti-Variante)

Dieses Feld belegt zwei Bits und dient zur Unterscheidung des Steuervektors von den 64 vordefinierten Varianten, die von Chiffriersystemen auf Variantenbasis verwendet werden. Da bei allen 64 vordefinierten Varianten alle acht Bytes identisch sind, kann ein gültiger Steuervektor von einer vordefinierten Variante unterschieden werden, indem mindestens zwei Bytes des Steuervektors verschieden gesetzt werden.

#### # Software-Bits



Dieses Feld belegt 12 Bit.

- I**

### Steuervektor für MAC-Schlüssel

Das Format der Steuervektoren für MAC-Schlüssel ist in Fig. 13 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- ```
# CV-Art - für MAC-Schlüssel (Hauptart = "Datenschlüssel",
Unterart = "MAC").
```

- # Exportkontrolle (regelt den Export dieses Schlüssels) - Hier gilt das gleiche wie für Vertraulichkeitsschlüssel.
- # Verwendung
- \* MG
  - MG = 1: Dieser Schlüssel darf in der Anweisung GMAC zur Generierung von MACS für Daten benutzt werden.
  - MG = 0: Dieser Schlüssel darf nicht in der Anweisung GMAC zur Generierung von MACs für Daten benutzt werden.
- \* MV
  - MV = 1: Dieser Schlüssel darf in der Anweisung VMAC zur Überprüfung von MACs für Daten benutzt werden.
  - MV = 0: Dieser Schlüssel darf nicht in der Anweisung VMAC zur Überprüfung von MACs für Daten benutzt werden.
- # AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.
- # Software-Bits - Dieses Feld belegt 12 Bit.
- # CV-Version - Wie bei Vertraulichkeitsschlüsseln.
- \* Durch Software erzwungene Verwendung - Siehe auch Abschnitt über das CFAP.
  - CVDML4 (Steuervektor Daten MACLEN = 4)
  - CVDM99 (Steuervektor Daten MAC MODE = ANSI X9.9)
  - CVDM19 (Steuervektor Daten MAC MODE = ANSI X9.19)
  - CVDM00 (Steuervektor Daten MAC MODE = IBM 4700)
  - CVDM30 (Steuervektor Daten MAC MODE = IBM 4730)
- # Länge - Wie bei Vertraulichkeitsschlüsseln.

# Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.

# Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektor für Datenkompatibilitätsschlüssel

Das Format der Steuervektoren für Datenkompatibilitätsschlüssel ist in Fig. 14 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

# CV-Art - für Datenkompatibilitätsschlüssel (Hauptart = "Datenschlüssel", Unterart = "Kompatibilität").

# Exportkontrolle - Wie bei Vertraulichkeitsschlüsseln.

# Verwendung

\* E

- E = 1: Dieser Schlüssel kann in der Anweisung "Chiffrieren" zum Chiffrieren von Daten benutzt werden.
- E = 0: Dieser Schlüssel kann nicht in der Anweisung "Chiffrieren" zum Chiffrieren von Daten benutzt werden.

\* D

- D = 1: Dieser Schlüssel kann in der Anweisung "Dechiffrieren" zum Dechiffrieren von Daten benutzt werden.
- D = 0: Dieser Schlüssel kann nicht in der Anweisung "Dechiffrieren" zum Dechiffrieren von Daten benutzt werden.

\* MG

- MG = 1: Dieser Schlüssel darf in der Anweisung GMAC zur Generierung von MACs für Daten benutzt werden.

- MG = 0: Dieser Schlüssel darf nicht in der Anweisung GMAC zur Generierung von MACs für Daten benutzt werden.
- \* MV
- MV = 1: Dieser Schlüssel darf in der Anweisung VMAC zur Überprüfung von MACs für Daten benutzt werden.
- MV = 0: Dieser Schlüssel darf nicht in der Anweisung VMAC zur Überprüfung von MACs für Daten benutzt werden.
- # AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.
- # Software-Bits - Dieses Feld belegt 12 Bit.
- # CV-Version - Wie bei Vertraulichkeitsschlüsseln.
- \* Software - Verwendung erzwungen
- # Länge - Wie bei Vertraulichkeitsschlüsseln.
- # Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.
- # Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektor für Datenumwandlungsschlüssel

Das Format der Steuervektoren für Datenumwandlungsschlüssel ist in Fig. 15 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- # CV-Art - für Datenumwandlungsschlüssel (Hauptart = "Datenschlüssel", Unterart = "XLATE").
- # Exportkontrolle (regelt den Export dieses Schlüssels) - Hier gilt das gleiche wie für Vertraulichkeitsschlüssel.

- # Verwendung
- \* XDout
  - XDout = 1: Dieser Schlüssel darf als Ausgabedatenschlüssel in der Anweisung "Chiffretext umwandeln" benutzt werden.
  - XDout = 0: Dieser Schlüssel darf nicht als Ausgabedatenschlüssel in der Anweisung "Chiffretext umwandeln" benutzt werden.
- \* XDin
  - XDin = 1: Dieser Schlüssel darf als Eingabedatenschlüssel in der Anweisung "Chiffretext umwandeln" benutzt werden.
  - XDin = 0: Dieser Schlüssel darf nicht als Eingabedatenschlüssel in der Anweisung "Chiffretext umwandeln" benutzt werden.
- # AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.
- # Software-Bits - Dieses Feld belegt 12 Bit.
- # CV-Version
- \* Durch Software erzwungene Verwendung - Keine.
- # Länge - Wie bei Vertraulichkeitsschlüsseln.
- # Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.
- # Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektoren für ANSI-Datenschlüssel

Das Format der Steuervektoren für ANSI-Datenschlüssel ist in Fig. 16 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- # CV-Art - für ANSI-Schlüssel (Hauptart = "Datenschlüssel", Unterart = "ANSI").
- # Exportkontrolle - Wie bei Vertraulichkeitsschlüsseln.
- # Verwendung
- \* E
  - E = 1: Dieser Schlüssel kann in der Anweisung "Chiffrieren" zum Chiffrieren von Daten benutzt werden.
  - E = 0: Dieser Schlüssel kann nicht in der Anweisung "Chiffrieren" zum Chiffrieren von Daten benutzt werden.
- \* D
  - D = 1: Dieser Schlüssel kann in der Anweisung "Dechiffrieren" zum Dechiffrieren von Daten benutzt werden.
  - D = 0: Dieser Schlüssel kann nicht in der Anweisung "Dechiffrieren" zum Dechiffrieren von Daten benutzt werden.
- \* MG
  - MG = 1: Dieser Schlüssel darf in der Anweisung GMAC zur Generierung von MACs für Daten benutzt werden.
  - MG = 0: Dieser Schlüssel darf nicht in der Anweisung GMAC zur Generierung von MACs für Daten benutzt werden.
- \* MV
  - MV = 1: Dieser Schlüssel darf in der Anweisung VMAC zur Überprüfung von MACs für Daten benutzt werden.

- MV = 0: Dieser Schlüssel darf nicht in der Anweisung VMAC zur Überprüfung von MACs für Daten benutzt werden.
- \* ACMB - Dieses Bit gibt an, ob der Datenschlüssel durch XOR mit einem anderen Datenschlüssel mit dem Attribut ACOMBKD verknüpft werden kann. Die XOR-Verknüpfung erfolgt durch die Anweisung ACOMBKD, wie im Abschnitt "ANSI Combine KDs (ACOMBKD)" beschrieben. Der resultierende Schlüssel dient zum Überprüfen und Generieren von MACs für die über das ANSI X9.17-Protokoll übertragenen Nachrichten.
- ACMB = 1: Dieser Datenschlüssel kann in der Anweisung ACMB verknüpft werden.
- ACMB = 0: Dieser Datenschlüssel kann nicht in der Anweisung ACMB verknüpft werden.
- # AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.
- # Software-Bits
- # CV-Version - Wie bei Vertraulichkeitsschlüsseln.
- \* Durch Software erzwungene Verwendung - Keine.
- # Länge - Wie bei Vertraulichkeitsschlüsseln.
- # Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.
- # Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektorformat für PIN-Schlüssel

PIN-Schlüssel sind in folgende Unterarten untergliedert:

- PIN-Chiffrierschlüssel (PEKs) - Mit diesen Schlüsseln werden PINs chiffriert.

- PIN-Generierungsschlüssel (PGKs) - Mit diesen Schlüsseln werden PINs generiert. Manchmal werden die PGKs auch als PIN-Prüf Schlüssel bezeichnet, da sie auch zum Überprüfen von PINs benutzt werden.

#### Steuervektoren für PIN-Chiffrierschlüssel

Das Format der Steuervektoren für PIN-Chiffrierschlüssel ist in Fig. 17 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- # CV-Art - für PIN-Chiffrierschlüssel (Hauptart = "PIN-Schlüssel", Unterart = "PIN-Chiffrierschlüssel").
- # Exportkontrolle - Wie bei Vertraulichkeitsschlüsseln.
- # Verwendung
- \* CREATE PINBLK
  - CREATE PINBLK = 1: Mit diesem Schlüssel kann der PIN-Block in der Anweisung "PIN-Block erstellen" chiffriert werden.
  - CREATE PINBLK = 0: Mit diesem Schlüssel darf der PIN-Block in der Anweisung "PIN-Block erstellen" nicht chiffriert werden.
- \* GENPIN
  - GENPIN = 1: Mit diesem Schlüssel kann die Eingabe-Kunden-identifikationszahl (CPIN) in der Anweisung "PIN generieren" chiffriert werden.
  - GENPIN = 0: Mit diesem Schlüssel darf die Eingabe-Kunden-identifikationszahl (CPIN) in der Anweisung "PIN generieren" nicht chiffriert werden.



\* VERPIN

- VERPIN = 1: Mit diesem Schlüssel kann die PIN-Eingabe an die Anweisung "PIN überprüfen" chiffriert werden.
- VERPIN = 0: Mit diesem Schlüssel darf die PIN-Eingabe an die Anweisung "PIN überprüfen" nicht chiffriert werden.

\* XPIN in

- XPIN in = 1: Mit diesem Schlüssel kann die Eingabe-PIN in der Anweisung "PIN umwandeln" chiffriert werden.
- XPIN in = 0: Mit diesem Schlüssel darf die Eingabe-PIN in der Anweisung "PIN umwandeln" nicht chiffriert werden.

\* XPIN out

- XPIN out = 1: Mit diesem Schlüssel kann die Ausgabe-PIN in der Anweisung "PIN umwandeln" chiffriert werden.
- XPIN out = 0: Mit diesem Schlüssel darf die Ausgabe-PIN in der Anweisung "PIN umwandeln" nicht chiffriert werden.

# AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.

# Software-Bits - Dieses Feld belegt 12 Bit.

# CV-Version - Wie bei Vertraulichkeitsschlüsseln.

\* Durch Software erzwungene Verwendung - Keine.

# Länge - Wie bei Vertraulichkeitsschlüsseln.

# Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.

# Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

Steuervektor für PIN-Generierungsschlüssel

Das Format der Steuervektoren für PIN-Generierungsschlüssel ist in Fig. 18 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- # CV-Art - für PIN-Chiffrierschlüssel (Hauptart = "PIN-Schlüssel", Unterart = "PIN-Chiffrierschlüssel").
- # Exportkontrolle - Wie bei Vertraulichkeitsschlüsseln.
- # Verwendung
- \* GENPIN - In diesem 2 Bit langen Feld wird angegeben, unter welchen Bedingungen der Schlüssel in der Anweisung "PIN generieren" zur Generierung von PINs oder PIN-Offsets verwendet werden darf.
  - GENPIN = B'00': nicht zur Generierung von PINs oder PIN-Offsets zugelassen.
  - GENPIN = B'01': zur Generierung einer PIN oder eines PIN-Offset im Klartext zugelassen.
  - GENPIN = B'10': zur Generierung einer chiffrierten PIN oder eines chiffrierten PIN-Offset zugelassen.
  - GENPIN = B'11': zur Generierung einer chiffrierten oder unchiffrierten PIN oder eines chiffrierten oder unchiffrierten PIN-Offset zugelassen.
- \* GPIN - Mit diesem Bit wird angegeben, ob die Kunden-PIN (CPIN) chiffriert oder im Klartext als Eingabe an die Anweisung GENPIN verwendet werden kann.
  - GPIN = 0: Chiffrierte oder unchiffrierte PIN zugelassen.
  - GPIN = 1: Nur chiffrierte PIN zugelassen.
- \* VERPIN - Mit diesem Bit wird angegeben, ob der Schlüssel als PIN-Prüfsschlüssel zur Überprüfung von PINs in der Anweisung "PIN überprüfen" benutzt werden darf.

- VERPIN = 1: Verwendung zur Überprüfung von PINs zugelassen.
- VERPIN = 0: Verwendung zur Überprüfung von PINs nicht zugelassen.
  
- \* VPIN - Mit diesem Bit wird angegeben, ob die in der Anweisung "PIN überprüfen" zu prüfende PIN im Klartext oder in chiffrierter Form an die Anweisung übergeben werden muß.
  
- VPIN = 0: Klartext oder chiffrierte PIN zugelassen.
- VPIN = 1: Nur chiffrierte PIN zugelassen.
  
- # AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln. Dieses Feld belegt 12 Bit.
  
- # Software-Bits
  
- \* CV-Version - Wie bei Vertraulichkeitsschlüsseln.
- \* Durch Software erzwungene Verwendung - Keine.
  
- # Länge - Wie bei Vertraulichkeitsschlüsseln.
  
- # Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.
  
- # Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektorformat für Schlüsselchiffrierschlüssel

Schlüsselchiffrierschlüssel sind in folgende Unterarten untergliedert:

- KEK (Schlüsselchiffrierschlüssel) Sender - Diese Art von Schlüsselchiffrierschlüsseln wird zum Senden oder Exportieren von Schlüsseln benutzt.
- KEK Empfänger - Diese Art von Schlüsselchiffrierschlüsseln wird zum Empfangen oder Importieren von Schlüsseln benutzt.

- KEK ANSI - Diese Art von Schlüsselchiffrierschlüsseln wird in einer Schlüsselverwaltungsumgebung nach ANSI X9.17 benutzt.

#### Steuervektorformat für KEK Sender

Das Format der Steuervektoren für KEK Sender ist in Fig. 19 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- # CV-Art - für KEK Sender (Hauptart = "Schlüsselchiffrierschlüssel", Unterart = "Sender").
- # Exportkontrolle - Wie bei Vertraulichkeitsschlüsseln.
- # Verwendung
- \* GKS - In diesem 3 Bit langen Feld wird festgelegt, ob der Schlüssel als Exportschlüssel in einem oder mehreren der folgenden Modi der Anweisung GKS verwendet werden kann: Import-Export, Lokal-Export und Export-Export.
- Bit 0: Import-Export-Modus. - In diesem Modus generiert die Anweisung GKS zwei Exemplare des Schlüssels. Ein Exemplar, das als Importexemplar bezeichnet wird, liegt in einer Form vor, in der es später vom generierenden Knoten importiert werden kann. Das andere Exemplar, das als Exportexemplar bezeichnet wird, liegt in einer Form vor, in der es an einen anderen Knoten gesendet werden kann.
  - Bit 0 = 1: Dieser Schlüssel ist für die Versendung des Exportexemplars des im IM-EX-Modus (Import-Export) generierten Schlüssels zugelassen.
  - Bit 0 = 0: Dieser Schlüssel ist nicht für die Versendung des Exportexemplars des im IM-EX-Modus (Import-Export) generierten Schlüssels zugelassen.

- Bit 1: Lokal-Export-Modus. - In diesem Modus generiert die Anweisung GKS zwei Exemplare des Schlüssels. Ein Exemplar des Schlüssels ist für die lokale Benutzung bestimmt (lokales Exemplar), das andere liegt in einer Form vor, in der es an einen anderen Knoten gesendet werden kann (Exportexemplar).
    - Bit 1 = 1: Dieser Schlüssel ist für die Versendung des Exportexemplars des im Op-Ex-Modus (Lokal-Export-Modus) generierten Schlüssels zugelassen.
    - Bit 1 = 0: Dieser Schlüssel ist nicht für die Versendung des Exportexemplars des im Op-Ex-Modus (Lokal-Export-Modus) generierten Schlüssels zugelassen.
  - Bit 2: Export-Export-Modus. - In diesem Modus generiert die Anweisung GKS zwei Exemplare des Schlüssels. Die beiden Exemplare liegen in einer Form vor, die an jeweils einen anderen Knoten versendet werden kann.
    - Bit 2 = 1: Dieser Schlüssel ist für die Versendung eines der beiden Exportexemplare des im Ex-Ex-Modus (Export-Export) generierten Schlüssels zugelassen. Das zu versendende Exemplar wird durch einen Parameter in der Anweisung GKS angegeben.
    - Bit 2 = 0: Dieser Schlüssel ist nicht für die Versendung von im Export-Export-Modus generierten Schlüsseln zugelassen.
- \* RFMK - Dieses Feld belegt ein Bit.
- RFMK = 1: Mit diesem Schlüssel dürfen Schlüssel mittels der Anweisung RFMK an anderen Knoten exportiert werden.
  - RFMK = 0: Mit diesem Schlüssel dürfen Schlüssel nicht mittels der Anweisung RFMK an andere Knoten exportiert werden.

\* XLTKEY out

- XLTKEY out = 1: Dieser Schlüssel kann in der Anweisung "Schlüssel umwandeln" als Schlüsselchiffrierschlüssel zur Umchiffrierung eines zuvor unter einem anderen Schlüssel chiffrierten Schlüssels verwendet werden.
- XLTKEY out = 0: Dieser Schlüssel darf nicht in der Anweisung "Schlüssel umwandeln" als Schlüsselchiffrierschlüssel zur Umchiffrierung eines zuvor unter einem anderen Schlüssel chiffrierten Schlüssels verwendet werden.
  
- # Schlüsselform - In diesem zwei Bit langen Feld wird angegeben, ob es sich um einen kurzen Schlüssel (einfache Länge) oder um einen langen Schlüssel (doppelte Länge) handelt, und ob es sich um die rechte oder um die linke Schlüsselhälfte handelt.
  
- # Verbindungssteuerung - In diesem zwei Bit langen Feld wird festgelegt, wie dieser KEK bei der Datenübertragung zum Senden oder Empfangen von Schlüsseln verwendet werden kann.
  
- VERBINDUNGSSTEUERUNG = B'00': entfällt.
- VERBINDUNGSSTEUERUNG = B'01': Nur CV. Das bedeutet, daß ein gesendeter Schlüssel K unter einem Schlüssel chiffriert sein muß, der durch XOR-Verknüpfung dieses KEK mit dem Steuervektor für den Schlüssel K gebildet wird. Diese Art der Verbindungsumgebung oder des Kanals ist offensichtlich für den Austausch von Schlüsseln geeignet, wenn in beiden beteiligten Knoten die CA implementiert ist.
- VERBINDUNGSSTEUERUNG = B'10': Nur CV = 0. Das bedeutet, daß ein gesendeter Schlüssel K unter diesem KEK chiffriert sein muß, und nicht unter dem Schlüssel, der durch XOR-Verknüpfung des KEK mit einem von Null verschiedenen Steuervektors gebildet wird.
- VERBINDUNGSSTEUERUNG = B'11': CV oder CV = 0. Dies bedeutet, daß ein gesendeter Schlüssel entweder nur unter dem KEK oder unter einem durch XOR-Verknüpfung des KEK mit dem Steuervektor für den Schlüssel K gebildeten Schlüssel chif-

friert sein kann. Diese Kanalart eignet sich für einen CA-Knoten, auf dem nicht nur Anwendungen in einer CA-Umgebung laufen, sondern auch ältere Anwendungen, die keine Steuervektoren erkennen. Bei älteren Anwendungen, die keine Steuervektoren erkennen, exportiert der Knoten Schlüssel, die nur unter dem KEK chiffriert sind. Bei Anwendungen, die in einer CA-Umgebung laufen, exportiert der Knoten Schlüssel, die unter dem durch XOR-Verknüpfung des KEK mit dem Steuervektor für den zu sendenden Schlüssel gebildeten Schlüssel chiffriert sind.

In dem Abschnitt über die Schlüsselverwaltung wird die Verwendung dieser Kanalarten zum Austauschen von Schlüsseln ausführlicher beschrieben.

- # AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.
- # Software-Bits - Dieses Feld belegt 12 Bit.
- # CV-Version - Wie bei Vertraulichkeitsschlüsseln.
- \* Durch Software erzwungene Verwendung - Siehe auch Abschnitt über das CFAP.
- # - CVKSKD (Steuervektor KEK Sender - Datenschlüssel senden)
- # - CVKSKP (Steuervektor KEK Sender - PIN-Schlüssel senden)
- # - CVKSKK (Steuervektor KEK Sender - KEK senden)
- # Länge - Wie bei Vertraulichkeitsschlüsseln.
- # Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.
- # Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektorformat für KEK Empfänger

Das Format der Steuervektoren für KEK Empfänger ist in Fig. 20 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- # CV-Art - für KEK Empfänger (Hauptart = "Schlüsselchiffrierschlüssel", Unterart = "Empfänger").
- # Exportkontrolle - Wie bei Vertraulichkeitsschlüsseln.
- # Verwendung
- \* GKS - In diesem 2 Bit langen Feld wird festgelegt, ob der Schlüssel als Exportschlüssel in einem oder mehreren der folgenden Modi der Anweisung GKS verwendet werden kann: Import-Export und Lokal-Import.
- Bit 0: Import-Export-Modus. - In diesem Modus generiert die Anweisung GKS zwei Exemplare des Schlüssels. Ein Exemplar, das als Importexemplar bezeichnet wird, liegt in einer Form vor, in der es später vom generierenden Knoten importiert werden kann. Das andere Exemplar, das als Exportexemplar bezeichnet wird, liegt in einer Form vor, in der es an einen anderen Knoten gesendet werden kann.
- Bit 0 = 1: Dieser Schlüssel ist zum Chiffrieren des Importexemplars des im IM-EX-Modus (Import-Export) generierten Schlüssels zugelassen.
  - Bit 0 = 0: Dieser Schlüssel ist nicht zum Chiffrieren des Importexemplars des im IM-EX-Modus (Import-Export) generierten Schlüssels zugelassen.
- Bit 1: Lokal-Export-Modus. - In diesem Modus generiert die Anweisung GKS zwei Exemplare des Schlüssels. Ein Exemplar des Schlüssels ist für die lokale Benutzung (lokales Exemplar) bestimmt, das andere liegt in einer Form vor, in der



es von einem anderen Knoten importiert werden kann (Import-exemplar).

- Bit 1 = 1: Dieser Schlüssel ist zur Chiffrierung des Importexemplars des im Op-Imp-Modus (Lokal-Import-Modus) generierten Schlüssels zugelassen.
- Bit 1 = 0: Dieser Schlüssel ist nicht zur Chiffrierung des Importexemplars des im Op-Imp-Modus (Lokal-Import-Modus) generierten Schlüssels zugelassen.

\* RTMK - Dieses Feld belegt ein Bit.

- RTMK = 1: Mit diesem Schlüssel dürfen Schlüssel mittels der Anweisung RTMK empfangen werden.
- RTMK = 0: Mit diesem Schlüssel dürfen Schlüssel nicht mittels der Anweisung RTMK empfangen werden.

\* XLTKEY in

- XLTKEY in = 1: Dieser Schlüssel kann zum Chiffrieren des Eingabeschlüssels für die Anweisung "Schlüssel umwandeln" verwendet werden.
- XLTKEY in = 0: Dieser Schlüssel darf nicht zum Chiffrieren des Eingabeschlüssels für die Anweisung "Schlüssel umwandeln" verwendet werden.

# Schlüsselform - In diesem zwei Bit langen Feld wird angegeben, ob es sich um einen kurzen Schlüssel (einfache Länge) oder um einen langen Schlüssel (doppelte Länge) handelt, und, falls der KEK ein langer Schlüssel ist, ob es sich um die rechte oder um die linke Schlüsselhälfte handelt.

# Verbindungssteuerung - Wie bei KEK Sender.

# AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.

- # Software-Bits - Dieses Feld belegt 12 Bit.
- # CV-Version - Wie bei Vertraulichkeitsschlüsseln.
- \* Durch Software erzwungene Verwendung - Siehe auch Abschnitt über das CFAP.
- CVKRKD (Steuervektor KEK Empfänger - Datenschlüssel empfangen)
- CVKRKP (Steuervektor KEK Empfänger - PIN-Schlüssel empfangen)
- CVKRKK (Steuervektor KEK Empfänger - KEK empfangen)
- # Länge - Wie bei Vertraulichkeitsschlüsseln.
- # Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.
- # Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektorformat für ANSI KEK

Das Format der Steuervektoren für ANSI KEK ist in Fig. 21 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- # CV-Art - (Hauptart = "Schlüsselchiffrierschlüssel (KEK)", Unterart = "ANSI").
- # Exportkontrolle - Wie bei Vertraulichkeitsschlüsseln.
- # Verwendung
- \* ARFMK - Gibt an, ob der Schlüssel in der Anweisung ARFMK als Schlüsselchiffrierschlüssel zum Versenden von Schlüsseln an andere Knoten verwendet werden kann.

- \* ARTMK - Gibt an, ob der Schlüssel in der Anweisung ARTMK als Schlüsselchiffrierschlüssel zum Empfangen von Schlüsseln von anderen Knoten verwendet werden kann.
- \* AXLTKEY - Gibt an, ob der Schlüssel in der Anweisung AXLTKEY als Schlüsselchiffrierschlüssel zur Umwandlung von Schlüsseln benutzt werden kann. Beim ANSI KEK-Steuervektor wird nicht zwischen Eingabe- und Ausgabe-Umwandlungsschlüsseln unterschieden, da diese Steuervektoren bidirektional ist.
- \* APNOTR - Gibt an, ob mit diesem Schlüssel Transformationen zur Erzeugung eines teilweise beglaubigten Schlüssels, mit dem andere Schlüssel beglaubigt werden können, möglich ist. Nach Abschluß der Transformation muß beim erzeugten teilweise beglaubigten Schlüssel KKPN dieses APNOTR-Bit auf Null zurückgesetzt sein, damit KKPN nicht als Eingabe für die Anweisung APNOTR benutzt werden kann, um einen anderen teilweise beglaubigten Schlüssel zu berechnen.
- # Schlüsselform - Wie bei KEK Sender.
- # Verbindungssteuerung - Dieses Feld entfällt bei ANSI KEK. Siehe auch KEK Sender.
- # AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.
- # Software-Bits - Dieses Feld belegt 12 Bit.
- # CV-Version - Wie bei Vertraulichkeitsschlüsseln.
- \* Durch Software erzwungene Verwendung - Keine.
- # Länge - Wie bei Vertraulichkeitsschlüsseln.
- # Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.

# Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektorformat für Schlüsselteile

Das Format der Steuervektoren für Schlüsselteile ist in Fig. 22 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

# CV-Art - CV-Art, wobei die drei letzten Bits xxx des Feldes den Wert Null haben, aber von den Anweisungen geprüft werden. (Hauptart = "Schlüsselteil", Unterart entfällt).

# Exportkontrolle - Wie bei Vertraulichkeitsschlüsseln.

# Schlüsselform - Wie bei KEK Sender.

# AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.

# Software-Bits - Dieses Feld belegt 12 Bit.

\* CV-Version - Wie bei Vertraulichkeitsschlüsseln.

\* Durch Software erzwungene Verwendung - Keine.

# Länge - Wie bei Vertraulichkeitsschlüsseln.

# Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.

# Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektorformat für Zwischen-ICV

Das Format der Steuervektoren für Zwischen-ICV ist in Fig. 23 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- # CV-Art - CV-Art, wobei die drei letzten Bits xxx des Feldes den Wert Null haben, aber von den Anweisungen geprüft werden. (Hauptart = "Zwischen-ICV", Unterart entfällt).
- # Exportkontrolle - Wie bei Vertraulichkeitsschlüsseln. Die ICV-Schlüssel werden normalerweise nicht an andere Knoten exportiert, außer bei einer Schnellsicherung. In diesem Feld steht deshalb bei ICV-Steuervektoren in der Regel der Wert B'00'.
- # AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.
- # Software-Bits
- \* CV-Version - Wie bei Vertraulichkeitsschlüsseln.
- \* Durch Software erzwungene Verwendung - Keine.
- # Länge - Wie bei Vertraulichkeitsschlüsseln.
- # Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.
- # Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Steuervektorformat für Tokens

Das Format der Steuervektoren für Tokens ist in Fig. 24 dargestellt. Im folgenden werden die Felder und Teilfelder dieser Tabelle ausführlich erläutert.

- # CV-Art - CV-Art, wobei die drei letzten Bits xxx des Feldes den Wert Null haben, aber von den Anweisungen geprüft werden. (Hauptart = "Token", Unterart entfällt). Tokens sind geheime Zeichenketten zum Schutz der Integrität der in der Datenschlüsseldatei (DKDS) gespeicherten Datenschlüssel. Datenschlüssel werden in der DKDS in der Form  
Token+e\*(KM.C1(K),e\*KM.C2(Token) gespeichert. Die Steuer-

vektorart für Tokens ist in verschiedenen Anweisungen zulässig, z.B. in den Anweisungen EMK, RTNMK und RTCMK.

- # Exportsteuerung - Wie bei Vertraulichkeitsschlüsseln. Die ICV-Schlüssel werden normalerweise nicht an andere Knoten exportiert, außer bei einer Schnellsicherung. In diesem Feld steht deshalb bei ICV-Steuervektoren in der Regel der Wert B'00'.
- # AV (Anti-Variante) - Wie bei Vertraulichkeitsschlüsseln.
- # Software-Bits
- \* CV-Version - Wie bei Vertraulichkeitsschlüsseln.
- \* Durch Software erzwungene Verwendung - Keine.
- # Länge - Wie bei Vertraulichkeitsschlüsseln.
- # Reservierte Bits - Wie bei Vertraulichkeitsschlüsseln.
- # Paritätsbit - Wie bei Vertraulichkeitsschlüsseln.

#### Anweisungssatz

Der hier beschriebene Anweisungssatz ist ein in der CF implementierter Satz gemeinsamer Chiffrierfunktionen. Dabei sind Ausnahmen möglich; MDC kann beispielsweise auf der Ebene des CFAP implementiert sein. Andere abweichende Implementierungen des Anweisungssatzes sollten nur nach sorgfältiger Abwägung der Sicherheitsanforderungen und der Produktumgebung realisiert werden.

Der Anweisungssatz basiert auf den Steuervektoren. Diese enthalten mehrere Felder, die in einem bestimmten System implementiert sein können, aber nicht müssen. Das hier beschriebene Mindestmaß der Steuervektorprüfung muß jedoch in jedem System in der CF

durchgeführt werden. Wenn die gesamte kryptographische Trennung nicht erforderlich ist oder wenn ein Teil des Anwendungssatzes für eine bestimmte Anwendung ausreicht, kann die Steuervektorprüfung für die nicht implementierten Funktionen ausgeschlossen werden. (Die Prüfung des Unterart-Feldes des Steuervektors, eines codierten Feldes, gewährleistet, daß Steuervektoren nicht durch ungültige Kombinationen in den Chiffrieranweisungen miteinander vermischt oder aneinander angeglichen werden können. Diese Prüfung ist entscheidend für die Sicherheit.)

Die Anweisungssatzgleichungen geben die für die Funktion erforderlichen Eingabedaten und die von der Funktion zu erwartenden Ausgabedaten für eine beschriebene Chiffrierfunktion an. Anstatt tatsächliche Daten an die Funktion zu übergeben, kann die Implementierung die Adressen der Daten an die Funktion übergeben. Diese Adressierung hängt von der jeweiligen Systemimplementierung ab und ist nicht Gegenstand dieser Beschreibung.

In den Gleichungen sind alle Parameter und Daten angegeben, die zur Ausführung einer Funktion benötigt werden. Je nach Arbeitsmodus werden in der Funktion alle oder nur einige Parameter verwendet. Die Felder in der Gleichung sind als Eingabe und Ausgabe zu verstehen, nicht als die tatsächlichen Werte der Ein- bzw. Ausgabedaten der Funktion in einer bestimmten Implementierung. Das Feld "A" in der Gleichung beispielsweise bezeichnet die Daten, die an die Funktion übergeben werden; die physische Form davon kann z.B. eine 32-Bit-Adresse sein, die auf die Daten verweist, welche von der Anweisung aus dem Speicher ausgelesen oder in den Speicher geschrieben werden sollen. Die Datenbusbreite vom Speicher hängt ebenfalls von der speziellen Implementierung ab, wohingegen die durch die Chiffrierfunktion zu chiffrierenden oder zu dechiffrierenden Eingabedaten immer ein Vielfaches von 8 Byte umfassen.

Für die Steuervektorprüfung gibt es zwei grundsätzliche Möglichkeiten:

1. Bits im Steuervektor prüfen: Es wird geprüft, ob die Steuervektorbits so gesetzt sind, wie sie gesetzt sein müssen. Bei Abweichungen wird ein Bedingungscode gesetzt und die Operation abgebrochen. Angenommen, in der Anweisung "Chiffrieren" wird das "E"-Bit des Steuervektors auf den Wert "1" geprüft. Wird festgestellt, daß dieses Bit nicht gleich "1" ist, wird die Operation abgebrochen. Bei komplizierteren Anweisungen ist diese Prüfung nicht so trivial, und es müssen Parameter übergeben werden, durch die der Umfang von Eingabe- und Ausgabeoperationen und die Verwendung des Steuervektors angegeben werden. In der Anweisung "Schlüsselsatz generieren" (GKS) beispielsweise wird für die Anweisung ein "Modus" angegeben, um eine bestimmte Form von Ausgabedaten zu erzeugen, und die Steuervektorprüfung muß entsprechend der Modusangabe erfolgen.
2. Bits im Steuervektor setzen: Bei diesem Verfahren wird ein Bit im Steuervektor der Operation entsprechend gesetzt und dann die Operation ausgeführt. Dabei müssen die Steuervektoren nicht geprüft werden. So kann z.B. in der Anweisung "Chiffrieren" das "E"-Bit im Steuervektor gesetzt werden, und daraufhin wird die Operation ausgeführt. In diesem Fall muß aber jede Anweisung wissen, welche Bits unter welchen Bedingungen gesetzt werden müssen. Diese Strategie funktioniert nicht in jedem Fall. Wie sollte die Anweisung beispielsweise wissen, wie das Bit für "Zielkontrolle" gesetzt werden muß? Dies bedeutet, daß der Anweisung ein Parameter übergeben werden muß, der der Anweisung mitteilt, daß sie einen bestimmten Bitwert im Steuervektor auswählen soll. Dadurch verliert das Steuervektorkonzept seine Flexibilität vollkommen.

Beide genannten Verfahren erfüllen die Anforderungen der Steuervektorprüfung. Wird haben uns aus folgenden Gründen für das



Verfahren "Bits im Steuervektor prüfen" entschieden (es werden keine Steuervektorbites von der Anweisung gesetzt):

1. Die Anweisung muß nicht wissen, welche Bits wann gesetzt werden müssen.
2. Es ist eine sehr flexible Lösung, einen Parameter wie z.B. "Modus" zu übergeben und alle möglichen Ausgabedatenkombinationen zu erhalten, und eine Kombination von Eingabedaten an die Anweisung zu übergeben.
3. Wenn die Bitwerte fest in der CF codiert sind, ist es sehr schwierig, die Architektur zu erweitern. Außerdem kann man kaum alle möglichen Kombinationen im voraus wissen.
4. Die Hardwareimplementierung ist einfacher, und es besteht eine größere Flexibilität für die Software.
5. Die den Steuervektoren zugrundeliegende Absicht wird bewahrt: Steuervektoren werden (derzeit) für die kryptographische Trennung und speziell für Sicherheitszwecke eingesetzt. Sie werden in der CA nicht zur "Angabe einer Operation oder Funktion für eine Anweisung" verwendet. Mit anderen Worten, Steuervektoren sind kein "erweiterter Operationscode" für die Anweisung.

#### Codieren (ENC)

|   |              |                                            |
|---|--------------|--------------------------------------------|
| E | - Gleichung: | KD, A -- eKD(A)                            |
|   | - Eingabe:   | KD            64-Bit-Schlüssel im Klartext |
|   |              | A            64 Bit Klartext               |
|   | - Ausgabe:   | eKD(A)      64 Bit chiffrierte Daten       |

Beschreibung: Die Codierfunktion wird zum Chiffrieren eines 8 Byte langen Klartext-Datenblocks im ECB-Modus mittels eines 64-Bit-Klartextschlüssels verwendet. An diese Anweisung wird kein Steuervektor übergeben.

Fig. 25 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. fehlgeschlagene Operation (Fehler)

HINWEIS: Eine fehlgeschlagene Operation kann irgendein für die betreffende Implementierung spezifischer Hardwarefehler sein. Die Bedingungscode (hier mit CC bezeichnet) sind nur ein Vorschlag; in einem System können auch mehr Bedingungscode implementiert sein. Die hier angegebenen CC-Codes sind ferner nicht die tatsächlichen Bedingungscode, die in einer bestimmten Implementierung von der Funktion übergeben werden müssen. Die Nummerierung dient zur übersichtlicheren Beschreibung der Chiffrierarchitektur.

# Steuervektorprüfung: - Keine.

#### Decodieren (DEC)

- Gleichung: KD, eKD(A) -- A
- Eingabedaten: KD 64-Bit-Schlüssel im Klartext  
eKD(A) 64 Bit chiffrierte Daten
- Ausgabedaten: A 64 Bit chiffrierte Daten

Beschreibung: Die Decodierfunktion wird zum Chiffrieren eines 8 Byte langen Klartext-Datenblocks im ECB-Modus mittels eines 64-Bit-Klartextschlüssels verwendet. An diese Anweisung wird kein Steuervektor übergeben.

Fig. 26 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung: - Keine.

### Chiffrieren (ENCI)

- Gleichung:  $e * KM.C1(KD1), ICV, A, n, C1 \rightarrow eKD1(ICV, A)$
  - Eingabe:  $e * KM.C1(KD1)$  64-Bit-Datenschlüssel (KD1), dreifach chiffriert unter dem Hauptschlüssel (KM) in Verbindung mit einem Steuervektor C1.
- ICV                      64 Bit chiffrierte Daten

HINWEIS: Chiffrierte ICVs werden vom CFAP verwaltet wie in den Abschnitten über die ICV/OCV-Verwaltung und über die Software-Schnittstelle beschrieben. Wenn ein Ausgabekettungswert (OCV) benötigt wird, müssen die letzten 8 Byte der Ausgabe (En) als OCV verwendet werden. Dies ist jedoch kein Standardverfahren. Die Chiffrierung und Dechiffrierung des letzten Blocks wird bei den einzelnen Systemimplementierungen unterschiedlich gehandhabt. Die Handhabung des letzten Blocks und die Verfahren zur OCV-Generierung werden im Abschnitt "Software-Schnittstelle" ausführlich beschrieben. Das CFAP verarbeitet jede mögliche Chiffrierung bzw. Dechiffrierung des letzten Blocks und auch die OCV-Verwaltung.

τ                      A                      Zu chiffrierende Daten in Vielfachen von 8-Byte-Blöcken

Die 8-Byte-Blöcke werden mit A1, A2, ..., An bezeichnet. Ist der letzte Block An kein Vielfaches von 8 Byte, muß dieser Block durch das CFAP aufgefüllt werden, bevor diese Anweisung aufgerufen wird. CF-Anweisungen gehen immer davon aus, daß die Länge der Ein- und Ausgabedaten ein Vielfaches von 8 Byte beträgt.

n                      Anzahl der zu chiffrierenden 8-Byte-Blöcke.

n sollte möglichst groß sein; die genaue Anzahl ist aber vom System abhängig. Die CA setzt keine Obergrenze für n. Beispiel: Wenn die Anzahl der 8-Byte-Blöcke = 10,000 und n max = 4,000 ist, wird die Chiffrieranweisung folgendermaßen aufgerufen:

```
-- n = 4000 chiffrieren
-- n = 4000 chiffrieren
-- n = 2000 chiffrieren
```

HINWEIS: Nach jedem Aufruf der Chiffrierfunktion müssen die letzten 8 Byte der chiffrierten Daten En(OCV) als ICV-Eingabe an den nächsten Chiffrierfunktionsaufruf übergeben werden.

```

c1          64-Bit-Steuervektor für Datenschlüssel
            (KD1).
-   Ausgabe: ekD1(ICV,A)   chiffrierte Daten, n Blöcke von
                           jeweils 8 Byte Länge (E1,
                           E2...En).
```

Beschreibung: Die Eingabedaten sind über den CBC-Modus der DEA-Chiffrierung verschlüsselt. Durch diese Anweisung werden Vielfache von 8-Byte-Blöcken chiffriert, bis alle n Blöcke chiffriert sind.

Die Architektur definiert nur unchiffrierte ICV-Eingaben an die Funktion. Wenn ein chiffrierter ICV benötigt wird, kann er mit der Chiffrieranweisung unter einem Datenschlüssel (KD2) chiffriert werden. Chiffrierte ICVs können mit der Dechiffrieranweisung entschlüsselt werden. Alle chiffrierten ICVs und OCVs werden vom CFAP-Programm verwaltet.

Ist der letzte Block An kein Vielfaches von 8 Byte, muß dieser Block durch das CFAP aufgefüllt werden, bevor die Chiffrieranweisung aufgerufen wird.

Fig. 27 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1 ist ungültig
- 3. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1

|               |   |                                                                             |
|---------------|---|-----------------------------------------------------------------------------|
| -- CV-Art     | = | "Daten/Kompatibilität" oder<br>"Daten/Vertraulichkeit" oder<br>"Daten/ANSI" |
| -- E-Bit      | = | 1                                                                           |
| -- reserviert | = | X'0'                                                                        |
|               |   | (48:63)                                                                     |

HINWEIS: Bei allen hier beschriebenen Anweisungen impliziert die Steuervektorprüfung, daß bei nichtbestandener Prüfung die Anweisung abgebrochen und der entsprechende Bedingungscode (CC) für einen ungültigen Steuervektor gesetzt wird. Wenn der Steuervektor einer oder mehreren Prüfungen unterzogen werden muß, werden alle Prüfungen durchgeführt, und alle müssen bestanden werden, damit die Operation durchgeführt wird. Wenn nur eine einzige Prüfung negativ ausfällt, muß die Operation abgebrochen werden.

#### Dechiffrieren (DECI)

- Gleichung:  $e * KM.C1(KD1), ICV, eKD1(ICV, A), n C1 \rightarrow A$
  - Eingabe:  $e * KM.C1(KD1)$  64-Bit-Datenschlüssel (KD1), dreifach chiffriert unter dem Hauptschlüssel (KM) in Verbindung mit einem Steuervektor C1.
- |     |                                                  |
|-----|--------------------------------------------------|
| ICV | 64 Bit langer unchiffrierter Eingabekettungswert |
|-----|--------------------------------------------------|

HINWEIS: Chiffrierte ICVs werden vom CFAP verwaltet wie in den Abschnitten "ICV/OCV-Verwaltung" und "Software-Schnittstelle" beschrieben. Wenn ein Ausgabekettungswert (OCV) benötigt wird, müssen die letzten 8 Byte der Ausgabe (En) als OCV verwendet werden. Dies ist jedoch kein Standardverfahren. Die Chiffrierung und Dechiffrierung des letzten Blocks wird bei den einzelnen Systemimplementierungen unterschiedlich gehandhabt. Die Handhabung des letzten Blocks und die Verfahren zur OCV-Generierung werden im Abschnitt "Software-Schnittstelle" ausführlich beschrieben. Das CFAP verarbeitet jede mögliche Chiffrierung bzw. Dechiffrierung des letzten Blocks und auch die OCV-Verwaltung.

|             |                                                                              |
|-------------|------------------------------------------------------------------------------|
| eKd1(ICV,A) | chiffrierte Daten, n Blöcke mit einer Länge von jeweils 8 Byte (E1, E2...En) |
| n           | Anzahl der zu chiffrierenden 8-Byte-Blöcke.                                  |

n sollte möglichst groß sein; die genaue Anzahl ist aber vom System abhängig. Die CA setzt keine Obergrenze für n.

HINWEIS: Nach jedem Aufruf der Dechiffrierfunktion müssen die letzten 8 Byte der chiffrierten Daten En(OCV) als ICV-Eingabe an den nächsten Dechiffrierfunktionsaufruf übergeben werden. Dies wird von CFAP erledigt. Beispiel: Wenn die Anzahl der 8-Byte-Blöcke = 10,000 und n max = 4,000 ist, wird die Dechiffrieranweisung folgendermaßen aufgerufen:

```
-- n = 4000 dechiffrieren
-- n = 4000 dechiffrieren
-- n = 2000 dechiffrieren
```

|              |                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------|
| C1           | 64-Bit-Steuervektor für Datenschlüssel (KD1).                                                             |
| - Ausgabe: A | Daten im Klartext in Vielfachen von 8-Byte-Blöcken. Die 8-Byte-Blöcke werden mit A1, A2,...An bezeichnet. |

Beschreibung: Die Eingabedaten sind über den CBC-Modus der DEA-Chiffrierung verschlüsselt. Durch diese Anweisung werden Vielfache von 8-Byte-Blöcken chiffriert, bis alle n Blöcke chiffriert sind.

Die Architektur definiert nur unchiffrierte ICV-Eingaben an die Funktion. Wenn ein chiffrierter ICV benötigt wird, kann er mit der Chiffrieranweisung unter einem Datenschlüssel (KD2) chiffriert werden. Chiffrierte ICVs können mit der Dechiffrieranweisung entschlüsselt werden. Alle chiffrierten ICVs und OCVs werden vom CFAP-Programm verwaltet.

Fig. 28 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1 ist ungültig
- 3. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1

|               |   |                                                                             |
|---------------|---|-----------------------------------------------------------------------------|
| -- CV-Art     | = | "Daten/Kompatibilität" oder<br>"Daten/Vertraulichkeit" oder<br>"Daten/ANSI" |
| -- D-Bit      | = | 1                                                                           |
| -- reserviert | = | X'0'                                                                        |
| (48:63)       |   |                                                                             |

#### MAC generieren (GMAC)

# Gleichung:  $e*KM.C1(KD2 = 1), [e*KM.C2(KD2)],$   
 $ICV[e*KM.C3(OCV)], A, n, ICV\text{-}Art, Ausgabeart,$   
 $mac\text{-}enc, C1, [C2], [C3] \text{ --- MAC (64 Bit)}$

oder

e\*KM.C3(OCV)

- Eingabe: e\*KM.C1(KD1)    KD1 ist ein MAC-Generierungsschlüssel für einfach chiffrierende MACs, dreifach chiffriert unter KM in Verbindung mit einem Steuervektor C1.

e\*KM.C2(KD2)    KD2 ist ein optionaler MAC-Generierungsschlüssel für dreifach chiffrierende MACs, dreifach chiffriert unter KM in Verbindung mit einem Steuervektor C2. Diese Eingabe ist nicht obligatorisch. Sie wird für eine dreifach chiffrierende MAC-Ausgabe benötigt, wenn mac-enc = 1 ist.

ICV    ICV = 0 ist der StandardICV-Wert für die CA-Architektur, ANSI X9.9 und ANSI X9.19. Von Null verschiedene unchiffrierte ICVs können ebenfalls verwendet werden, um die Kompatibilität mit Systemen zu wahren, die eine unchiffrierte ICV-Eingabe benötigen. Eine chiffrierte ICV-Eingabe wird von der CA nicht unterstützt, da es sich erwiesen hat, daß dadurch die Sicherheit für die Funktion nicht erhöht wird. Chiffrierte Zwischen-ICVs werden von der CA unterstützt.

HINWEIS: Chiffrierte ICVs werden bei Bedarf durch das CFAP verwaltet wie in den Abschnitten "ICV/Verwaltung" und "Software-Schnittstelle" beschrieben.



e\*KM.C3(OCV) Dies ist ein 64 Bit langer Zwischen-ICV, der unter dem Hauptschlüssel in Verbindung mit einem speziellen Steuervektor C3 chiffriert ist. Diese Eingabe ist nur erforderlich, wenn zur MAC-Generierung große Datenblöcke verwendet werden (/n). Die Dechiffrierung des ICV erfolgt funktionsintern. Zwischen-OCV können nicht vom lokalen Knoten versendet werden, da sie unter dem Hauptschlüssel in Verbindung mit einem Steuervektor in einer Form gespeichert sind, die nur für die lokale Verwendung bestimmt ist.

A Daten für die MAC-Operation in Vielfachen von 8-Byte-Blöcken (A1, A2...An).

n Anzahl der 8-Byte-Blöcke für die MAC-Operation.

n sollte möglichst groß sein; die genaue Anzahl ist aber vom System abhängig. Die CA setzt keine Obergrenze für n. Bei MAC-Operationen mit vielen Datenblöcken wird GMAC mehrmals aufgerufen, bis alle Blöcke verarbeitet sind. Beispiel: Angenommen, n max für das System beträgt 4000 und die Daten für die MAC-Operation umfassen 10,000 Blöcke von jeweils 8 Byte Länge, so wird GMAC folgendermaßen aufgerufen:

```
-- GMAC n = 4000,Ausgabeart = 1,ICV-Art = 0
-- GMAC n = 4000,Ausgabeart = 1,ICV-Art = 2
-- GMAC n = 2000,Ausgabeart = 1,ICV-Art = 2
```

HINWEIS: Nach jedem Aufruf von GMAC muß der Zwischen-ICV e\*KM.C3(OCV) dem nächsten GMAC-Aufruf als ICV-Eingabe übergeben

werden. Die Dechiffrierung dieses Zwischen-ICV muß in der CF intern erfolgen.

ICV-Art      Die ICV-Art gibt an, ob der an die Funktion übergebene ICV ein Null-ICV, ein Klartext-ICV oder ein Zwischen-ICV ist.

Zwischen-ICVs sind unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C3 dreifach chiffriert. Standardmäßig ist "Null-ICV" vorgegeben.

- 0: Null-ICV
- 1: Klartext-ICV
- 2: Zwischen-ICV (OCV)

Ausgabeart      Dieser Parameter teilt der Anweisung das Stadium des MAC-Generierungsprozesses mit.

- 0: MAC-Ausgabe
- 1: Zwischen-ICV-Ausgabe (OCV)

mac-enc      mac-enc gibt an, ob es sich um eine einfach chiffrierte oder um eine dreifach chiffrierende MAC-Ausgabe handelt.

- 0: einfach chiffrierende MAC-Ausgabe
- 1: dreifach chiffrierende MAC-Ausgabe (nach ANSI 9.19 erforderlich)

C1,C2,C3      64-Bit-Steuervektoren für KD1, KD2 und OCV. C2 und C3 sind nicht obligatorisch für die Anweisung. C2 muß angege-

ben werden, wenn mac-enc = 1 ist; C3 muß für ICV-Art = 2 oder für Ausgabeart = 1 angegeben werden.

- Ausgabe:

MAC

64-Bit-Ausgabe als Ergebnis einfacher oder dreifacher Chiffrierung des letzten Eingabedatenblocks, je nachdem, was unter dem Parameter mac-enc angegeben wurde. Diese Ausgabe ist nur für Ausgabeart = 0 gültig.

e\*KM.C3(OCV)

OCV ist ein 64 Bit langer Zwischen-ICV, der unter KM in Verbindung mit dem Steuervektor C3 dreifach chiffriert ist.

Diese Ausgabe ist nur für Ausgabeart = 0 gültig. MAC-Ausgabe und Zwischen-OCV-Ausgaben dürfen aus Sicherheitsgründen nicht gleichzeitig ausgegeben werden.

☛ Beschreibung: Die Eingabedaten sind über den CBC-Modus der DEA-Chiffrierung verschlüsselt und der letzte Block der chiffrierten Daten wird ausgegeben. Es gibt zwei Modi, nämlich die einfache Chiffrierung und die dreifache Chiffrierung. Bei der einfachen Chiffrierung wird zur MAC-Erstellung ein einziger Schlüssel KD1 verwendet. Bei der dreifachen Chiffrierung erfolgt zuerst eine einfache Chiffrierung mit KD1 zur MAC-Erstellung, dann wird der MAC mit KD2 dechiffriert und anschließend wieder mit KD1 chiffriert, um den endgültigen 64-Bit-MAC zu erzeugen.

Die Anweisung erzeugt eine 64 Bit lange MAC-Ausgabe. X9.9 verwendet allerdings eine 32 Bit lange MAC-Ausgabe, die aus der

linken Hälfte der 64 Bit langen MAC-Ausgabe besteht. Das CFAP muß die entsprechenden MAC-Bits aus der 64 Bit langen MAC-Ausgabe abtrennen.

ICV ist standardmäßig Null und kann wahlweise als Klartext- oder Zwischen-ICV an die Anweisung GMAC übergeben werden. Wenn chiffrierte ICVs erforderlich sind, muß das CFAP die ICVs unter (KDS) chiffrieren und einen Klartext-ICV an die Anweisung GMAC übergeben. Der MAC-Standard nach dem ANSI X9.9-Protokoll sieht einen Null-ICV für den ersten Block vor und ist deshalb hier als Standardeingabe definiert. Die Architektur ermöglicht aber auch Klartext- und Zwischen-ICV-Eingaben, um allen möglichen Bedürfnissen bei der MAC-Generierung Rechnung zu tragen.

Ist eine MAC-Generierung für Blöcke erforderlich, die größer als n sind, muß zur MAC-Generierung die Zwischen-ICV-Option verwendet werden. Dies bietet zusätzliche Sicherheit, da Zwischen-ICVs nicht unchiffriert vorliegen.

Ist der Datenblock kein Vielfaches von 8 Byte, muß er auf das nächste Vielfache von 8 Byte aufgefüllt werden. Diese Auffüllung muß vor dem Funktionsaufruf vom CFAP durchgeführt werden. Die MAC-Berechnung muß für Binärdaten gemäß ANSI X9.9-1986 Abschnitt 5.0 erfolgen, und eine eventuell notwendige Identifikationsüberprüfung muß vom CFAP implementiert werden.

Fig. 29 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1 ist ungültig
- 3. C2 oder C3 ist ungültig
- 4. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1
  - CV-Art = "Daten/Kompatibilität",  
"Daten/MAC" oder "Daten/ANSI"
  - MG-Verwendungsbit = 1
  - reserviert = X'0'  
(48:63)
- 2. Prüfung von C2
  - CV-Art = "Daten/Kompatibilität",  
"Daten/MAC" oder "Daten/ANSI"
  - MG-Verwendungsbit = 1
  - reserviert = X'0'  
(48:63)
- 3. Prüfung von C3, falls (ICV-Art = 2 ODER Ausgabeart = 1).
  - CV-Art = "Zwischen-ICV"

#### MAC-Überprüfung (VMAC)

- # Gleichung:  $e*KM.C1(KD1), [e*KM.C2(KD2)],$   
 $ICV[e*KM.C3(OCV)], A, MAC, n, ICV-Art,$   
 Ausgabeart, mac-enc, mac-len, C1, [C2], [C3]
- ja/nein  
oder  
e\*KM.C3(OCV)
- Eingabe:  $e*KM.C1(KD1)$  KD1 ist ein MAC-Prüf Schlüssel  
für einfach chiffrierende  
MACs, dreifach chiffriert  
unter KM in Verbindung mit  
einem Steuervektor C1.

e\*KM.C2(KD2)     KD2 ist ein optionaler MAC-Prüf Schlüssel für dreifach chiffrierende MACs, dreifach chiffriert unter KM in Verbindung mit einem Steuervektor C2.

Diese Eingabe ist nicht obligatorisch. Sie wird für eine dreifach chiffrierende MAC-Ausgabe benötigt, wenn mac-enc = 1 ist.

ICV                     ICV = 0 ist der StandardICV-Wert für die CA-Architektur, ANSI X9.9 und ANSI X9.19.

Von Null verschiedene unchiffrierte ICVs können ebenfalls verwendet werden, um die Kompatibilität mit Systemen zu wahren, die eine unchiffrierte ICV-Eingabe benötigen. Eine chiffrierte ICV-Eingabe wird von der CA nicht unterstützt, da es sich erwiesen hat, daß dadurch die Sicherheit für die Funktion nicht erhöht wird. Chiffrierte Zwischen-ICVs werden von der CA unterstützt.

HINWEIS: Chiffrierte ICVs werden bei Bedarf durch das CFAP verwaltet wie in den Abschnitten "ICV/Verwaltung" und "Software-Schnittstelle" beschrieben.

τ

e\*KM.C3(OCV)     Dies ist ein 64 Bit langer Zwischen-ICV, der unter dem Hauptschlüssel in Verbindung mit einem speziellen Steuervektor C3 chiffriert ist.

Diese Eingabe ist nur erforderlich, wenn zur MAC-Überprüfung große Datenblöcke verwendet werden (] n). Die Dechiffrierung des ICV erfolgt funktionsintern. Zwischen-OCV können nicht vom lokalen Knoten versendet werden, da sie unter dem Hauptschlüssel in Verbindung mit einem Steuervektor in einer Form gespeichert

sind, die nur für die lokale Verwendung bestimmt ist. Die Zwischen-ICVs müssen bei der MAC-Überprüfung geheim sein, damit kein Angriff auf die Sicherheit möglich ist.

|     |                                                                                         |
|-----|-----------------------------------------------------------------------------------------|
| A   | Daten für die MAC-Operation<br>in Vielfachen von 8-Byte-<br>Blöcken (A1, A2...An).      |
| MAC | 64 Bit umfassende MAC-Eingabe<br>an die Anweisung, einfach<br>oder dreifach chiffriert. |

Standardmäßig werden nur die letzten 32 Bit dieses MAC für den MAC-Vergleich herangezogen. Mit mac-len können explizit andere Längen für den Vergleich angegeben werden.

|   |                                                    |
|---|----------------------------------------------------|
| n | Anzahl der 8-Byte-Blöcke für<br>die MAC-Operation. |
|---|----------------------------------------------------|

n sollte möglichst groß sein; die genaue Anzahl ist aber vom System abhängig. Bei der MAC-Überprüfung vieler Datenblöcke wird VMAC mehrmals aufgerufen, bis alle Blöcke verarbeitet sind. Beispiel: Angenommen, n max für das System beträgt 4000 und die zu überprüfenden umfassen 10,000 Blöcke von jeweils 8 Byte Länge, so wird VMAC folgendermaßen aufgerufen:

```
-- VMAC n = 4000,Ausgabeart = 1,ICV-Art = 0
-- VMAC n = 4000,Ausgabeart = 1,ICV-Art = 2
-- VMAC n = 2000,Ausgabeart = 0,ICV-Art = 2
```

HINWEIS: Nach jedem Aufruf von VMAC muß der Zwischen-ICV e\*KM.C3(OCV) dem nächsten VMAC-Aufruf als ICV-Eingabe übergeben werden. Die Dechiffrierung dieses Zwischen-ICV muß in der CF intern erfolgen.

|         |                                                                         |
|---------|-------------------------------------------------------------------------|
| ICV-Art | Die ICV-Art gibt an, ob der an die<br>Funktion übergebene ICV ein Null- |
|---------|-------------------------------------------------------------------------|

ICV, ein Klartext-ICV oder ein Zwischen-ICV ist.

Zwischen-ICVs sind unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C3 dreifach chiffriert.

- 0: Null-ICV (Standardwert)
- 1: Klartext-ICV
- 2: Zwischen-ICV (OCV)

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| Ausgabeart | Dieser Parameter teilt der Anweisung das Stadium des MAC-Überprüfungsprozesses mit. |
|------------|-------------------------------------------------------------------------------------|

- 0: MAC-Überprüfungsausgabe
- 1: Zwischen-ICV-Ausgabe (OCV)

|         |                                                                                                                  |
|---------|------------------------------------------------------------------------------------------------------------------|
| mac-enc | mac-enc gibt an, ob es sich um eine einfach chiffrierte oder um eine dreifach chiffrierende MAC-Eingabe handelt. |
|---------|------------------------------------------------------------------------------------------------------------------|

- 0: einfach chiffrierende MAC-Eingabe
- 1: dreifach chiffrierende MAC-Eingabe (nach ANSI 9.19 erforderlich)

|         |                                                                   |
|---------|-------------------------------------------------------------------|
| mac-len | mac-len gibt an, wieviele Bytes des MAC verglichen werden sollen. |
|---------|-------------------------------------------------------------------|

Standardmäßig werden die ersten vier Bytes von links verglichen.

- 0: 4 Bytes von links
- 1: 5 Bytes von links
- 2: 6 Bytes von links



- 3: 7 Bytes von links
- 4: 8 Bytes

HINWEIS: Die Möglichkeit, 4, 5, 6, 7 oder 8 Byte für die MAC-Überprüfung auszuwählen, kann die MAC-Generierung für 8-Byte-MACs stören. Die Lösung dieses Problems ist nicht Bestandteil der vorliegenden Erfindung und wird deshalb nicht weiter erläutert.

C1,C2,C3            64-Bit-Steuervektoren für  
KD1, KD2 und OCV.

C2 und C3 sind nicht obligatorisch für die Anweisung. C2 muß angegeben werden, wenn mac-enc = 1 ist; C3 muß für ICV-Art = 2 oder für Ausgabeart = 1 angegeben werden.

- Ausgabedaten:    ja/nein            Gibt an, ob das MAC überprüft wird.
- e\*KM.C3(OCV)        OCV ist ein 64 Bit langer Zwischen-ICV, der unter KM in Verbindung mit dem Steuervektor C3 dreifach chiffriert ist. Diese Angabe kann nur für Ausgabeart = 1 gemacht werden.

Beschreibung: Die Eingabedaten werden über den CBC-Modus der DEA-Chiffrierung mittels des Datenschlüssels KD1 verschlüsselt, und die linken 32 Bits des letzten chiffrierten Blocks werden auf Übereinstimmung mit dem übergebenen MAC geprüft. MAC-Vergleich ist ein Standardwert, und andere Vergleiche müssen der Angabe durch den Parameter mac-len entsprechend durchgeführt werden. Bei Übereinstimmung der MACs wird CC = 1 gesetzt, bei Nichtübereinstimmung CC = 2. Es gibt zwei Chiffriermodi, nämlich die einfache Chiffrierung und die dreifache Chiffrierung. Bei der einfachen Chiffrierung wird zur MAC-Erstellung ein einziger

Schlüssel KD1 verwendet. Bei der dreifachen Chiffrierung erfolgt zuerst eine einfache Chiffrierung mit KD1 zur MAC-Erstellung, dann wird das MAC mit KD2 dechiffriert und anschließend wieder mit KD1 chiffriert, um das endgültige 64-Bit-MAC zu erzeugen. Das MAC wird der mac-enc-Eingabe entsprechend generiert und anschließend mit dem der Funktion übergebenen MAC verglichen.

Der erste ICV ist standardmäßig ein Null-ICV. Optional kann ein Klartext-ICV verwendet werden. Wenn die Daten für die MAC-Operation mehr als n Blöcke umfassen, müssen Zwischen-ICVs verwendet werden.

Ist der Datenblock kein Vielfaches von 8 Byte, muß er auf das nächste Vielfache von 8 Byte aufgefüllt werden. Diese Auffüllung muß vor dem Funktionsaufruf vom CFAP durchgeführt werden.

Fig. 30 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. MACs stimmen überein
- 2. MACs stimmen nicht überein
- 3. C1 ist ungültig
- 4. C2 oder C3 ist ungültig
- 5. fehlgeschlagene Operation (Fehler)

£

# Steuervektorprüfung:

- 1. Prüfung von C1

|                      |   |                                                               |
|----------------------|---|---------------------------------------------------------------|
| -- CV-Art            | = | "Daten/Kompatibilität",<br>"Daten/MAC" oder "Da-<br>ten/ANSI" |
| -- MG-Verwendungsbit | = | 1                                                             |
| -- reserviert        | = | X'0'                                                          |
| (48:63)              |   |                                                               |

- 2. Prüfung von C2 falls (mac-enc = 1).
  - CV-Art = "Daten/Kompatibilität",  
"Daten/MAC" oder "Daten/ANSI"
  - MG-Verwendungsbit = 1
  - reserviert = X'0'  
(48:63)
- 3. Prüfung von C3, falls (ICV-Art = 2 ODER Ausgabeart = 1).
  - CV-Art = "Zwischen-ICV"

## Chiffretext umwandeln (TCTXT)

- ```

- Gleichung:      e*KM.C1(KD1),ICV1,eKD1(ICV1,A),
                  e*KM.C2(KD2),ICV2, n, C1, C2
                  ---- eKD2(ICV2,A)

- Eingabe:      e*KM.C1(KD1)    KD1 ist ein Eingabedaten-
                                schlüssel, dreifach chiff-
                                riert unter KM in Verbindung
                                mit einem Steuervektor C1.
                                ICV1    unchiffrierter 64-Bit-ICV.
                                eKD1(ICV1,A)    KD2 ist ein Ausgabedaten-
                                                schlüssel, dreifach chiff-
                                                riert unter KM in Verbindung
                                                mit einem Steuervektor C2.
                                ICV2    unchiffrierter 64-Bit-ICV.
                                n        Anzahl der umzuwandelnden 8-
                                                Byte-Datenblöcke.
                                C1, C2    Steuervektoren für KD1 bzw.
                                                KD2.

- Ausgabe:      eKD2(ICV2,A)    Ausgabe von Daten A, chiff-
                                riert mit dem Datenschlüssel
                                KD2 unter Verwendung von
                                ICV2.

```

**Beschreibung:** Die Anweisung "Chiffretext umwandeln" chiffriert Daten von einem Datenschlüssel und ICV unter einen anderen Datenschlüssel und ICV um. Diese Anweisung kann für Schlüssel der Arten "Daten/XLT" und "Daten/Kompatibilität" verwendet werden. Zur Umwandlung von Daten können CV-Schlüssel oder CV=0-Schlüssel verwendet werden; die Schlüsselarten dürfen jedoch nicht gemischt oder abgeglichen werden. Die Daten können aus bis zu n 8-Byte-Blöcken bestehen, und die Umwandlung erfolgt in der Chiffriervorrichtung, so daß die unchiffrierten Daten nicht außerhalb der Chiffriervorrichtung offengelegt werden.

Die ICV-Eingaben ICV1 und ICV2 dürfen nur unchiffrierte ICV-Eingaben für die Anweisung sein. Zwischen-ICVs werden von der Funktion nicht erzeugt. Müssen mehr als n Datenblöcke umgewandelt und die Daten verkettet werden, muß das CFAP den letzten 8 Byte langen Chiffredatenblock (En) als Eingabe an ICV übergeben. Bei Verwendung chiffrierter ICVs muß das CFAP die ICVs vor der Übergabe an die Anweisung zuerst dechiffrieren.

HINWEIS: Die Anweisung "Chiffretext umwandeln" wurde speziell für Schlüssel mit dem Attribut "nur CV" (d.h. Datenumwandlungsschlüssel) entwickelt. Ein Kompatibilitätsmodus existiert nicht. Um dieses Problem zu umgehen, akzeptiert "Chiffretext umwandeln" Datenschlüssel mit D- und E-Attribut sowie XDin- und XDout-Attribut. Dies ist ein Service, der die Gefahr einer versehentlichen Offenlegung unchiffrierter Daten verringern soll, da ein Insider mit bösen Absichten die Daten mit Hilfe der Anweisung "Dechiffrieren" unchiffriert rekonstruieren kann. Die Schlüssel zur Umwandlung von Chiffretext können im Kompatibilitätsmodus nicht isoliert werden, und die Erstellung eines Steuervektors hierfür würde falsche Sicherheit vorspiegeln, wenn in Wirklichkeit mittels RTMK ein Angriff durch Importieren eines als Umwandlungsschlüssels vorgesehenen Schlüssels als Dechiffrierschlüssel (über den CV=0-Kanal) erfolgen kann. Der Kanal für die Umwandlung von Chiffretext könnte durch die Dechiffrierung der über diesen Kanal gesendeten Daten untergraben werden.

Fig. 31 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1 oder C2 ist ungültig
- 3. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1
  - CV-Art = "Daten/XLT" oder "Daten/Kompatibilität"
  - Wenn (CV-Art = "Daten/XLT", dann XDin = 1
  - reserviert (48:63) = X'0'
- 2. Prüfung von C2, falls (mac-enc = 1)
  - CV-Art = "Daten/XLT" oder "Daten/Kompatibilität"
  - Wenn (CV-Art = "Daten/XLT", dann XDout = 1
  - reserviert (48:63) = X'0'
- 3. Prüfung von C1 und C2
  - CV-Art (C1) = CV-Art (C2)

Schlüsselsatz generieren (GKS)

- Gleichung: OP-OP-Modus Modus, C3, C4 --- e\*KM.C3(K),  
e\*KM.C4(K)
- Gleichung: OP-EX-Modus Modus, C2L, C2R, C3, C4,  
e\*KM.C2L(KEK2L), e\*KM.C2R(KEK2R)  
--- e\*KM.C3(K), e\*KEK2.C4(K)
- Gleichung: EX-EX-Modus Modus, C1L, C1R, C2L, C2R, C3, C4,  
e\*KM.C1L(KEK1L), e\*KM.C1R(KEK1R),  
e\*KM.C2L(KEK2L), e\*KM.C2R(KEK2R)  
--- e\*KEK1.C3(K), e\*KEK2.C4(K)
- Gleichung: OP-IM-Modus Modus, C2L, C2R, C3, C4,  
e\*KM.C2L(KEK2L), e\*KM.C2R(KEK2R)  
--- e\*KM.C3(K), e\*KEK2.C4(K)

- Gleichung: IM-EX-Modus    Modus, C1L, C1R, C2L, C2R, C3, C4,  
                                  e\*KM.C1L(KEK1L), e\*KM.C1R(KEK1R),  
                                  e\*KM.C2L(KEK2L), e\*KM.C2R(KEK2R)  
                                  --- e\*KM.C3(K), e\*KEK2.C4(K)
- Eingabe:

Modus            gibt Ein- und Ausgabeformat der  
 Anweisung GKS sowie Schlüsselgene-  
 rierungsmodi an.

Die Schlüsselgenerierungsmodi werden im Abschnitt über die  
 Schlüsselverwaltung ausführlicher beschrieben.

- 0: OP-OP (lokal - lokal)
- 1: OP-EX (lokal - Export)
- 2: EX-EX (Export - Export)
- 3. OP-IM (lokal - Import)
- 4. IM-EX (Import - Export)

C1L,C1R        64-Bit-Steuervektoren, linker und  
 rechter Steuervektor für 128-Bit-  
 KEK1.

C1L ist der Steuervektor für KEK1L und C1R der Steuervektor für  
 KEK1R. Die tatsächliche Implementierung kann auch nur ein CV  
 übergeben und die Schlüsselformbits im Steuervektor implizit von  
 der CF setzen lassen. Bei der CA wurde entschieden, für KEKs den  
 rechten und den linken Steuervektor separat zu übergeben. Diese  
 beiden Steuervektorarten für die KEKs müssen vom CFAP generiert  
 und verwaltet werden.

C2L,C2R        64-Bit-Steuervektoren, linker und  
 rechter Steuervektor für 128-Bit-  
 KEK2.

C2L ist der Steuervektor für KEK2L und C2R der Steuervektor für KEK2R.

C3,C4	64-Bit-Steuervektoren für den Ausgabeschlüssel.
e*KM.C1L(KEK1L)	KEK1L ist ein linker 64-Bit-Schlüsselchiffrierschlüssel, der die linke Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels KEK1 bildet, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C1L.
e*KM.C1R(KEK1R)	KEK1R ist ein rechter 64-Bit-Schlüsselchiffrierschlüssel, der die rechte Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels KEK1 bildet, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C1R.
e*KM.C2L(KEK2L)	KEK2L ist ein linker 64-Bit-Schlüsselchiffrierschlüssel, der die linke Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels KEK2 bildet, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C2L.
e*KM.C2R(KEK2R)	KEK2R ist ein rechter 64-Bit-Schlüsselchiffrierschlüssel, der die rechte Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels KEK2 bildet, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C2R.



HINWEIS: Aus Gründen der Kompatibilität unterstützt die Architektur sowohl Schlüsselchiffrierschlüssel (KEKs) einfacher Länge (64 Bit) als auch Schlüsselchiffrierschlüssel (KEKs) doppelter Länge. Alle KEKs werden jedoch als 128-Bit-Schlüssel im CKDS (Schlüsselspeicher) gespeichert. Ein echter Schlüssel doppelter Länge  $KEK = KEK_{links} \parallel KEK_{rechts}$  (wobei  $KEK_{links}$  die linken 64 Bit und  $KEK_{rechts}$  die rechten 64 Bit von KEK sind), wird zur Speicherung dreifach unter dem Hauptschlüssel  $*KM$  in Verbindung mit einem Steuervektor  $LC$  und  $CR$  chiffriert. Der gespeicherte Schlüssel ist 128 Bit lang und hat die Form  $e*KM.C(KEK) = e*KM.CL(KEK_{links}) \parallel e*KM.CR(KEK_{rechts})$ . Ein KEK einfacher Länge wird durch Verdoppelung zu einem 128-Bit-Schlüssel;  $KEK \parallel KEK$  wird dann dreifach unter  $*KM$  und einem Steuervektor  $CL, CR$  chiffriert. Der gespeicherte Schlüssel ist 128 Bit lang und hat die Form  $e*KM.C(KEK \parallel KEK) = e*KM.CL(KEK) \parallel e*KM.CR(KEK)$ . Die Schlüsselformbits des verdoppelten KEK-Steuervektors haben in beiden Steuervektoren  $CL$  und  $CR$  den Wert "01", so daß der Schlüssel an einem Zielknoten, an dem KEK nur in 64-Bit-Form gespeichert ist, rekonstruiert werden kann. Die Schlüsselformbits für normale KEK-Steuervektoren haben für  $CL$  den Wert "10" und für  $CR$  den Wert "11", wodurch eine vollständige Trennung erreicht wird und ein Arbeitsfaktor von  $2^{*112}$  bewahrt wird.

τ - Ausgabe:

$e*KM.C3(K)$	K ist ein zufallsmäßig generierter 64-Bit-Schlüssel, dreifach chiffriert unter $KM$ in Verbindung mit einem Steuervektor $C3$ .
--------------	---

Der Steuervektor  $C3$  wird immer benutzt. Der Schlüssel  $\tilde{K}$  wird mit ungerader Parität versehen. Dieser Schlüssel kann im Knoten intern benutzt werden.

$e*KM.C4(K)$  K ist ein zufallsmäßig generierter 64-Bit-Schlüssel, dreifach chiffriert unter KM in Verbindung mit einem Steuervektor C4.

Der Steuervektor C4 wird immer benutzt. Der Schlüssel K wird mit ungerader Parität versehen. Dieser Schlüssel kann im Knoten intern benutzt werden.

$e*KEK1.C3(K)$  K ist ein zufallsmäßig generierter 64-Bit-Schlüssel, chiffriert unter KEK1 in Verbindung mit einem Steuervektor C3.

Der Steuervektor C3 wird immer benutzt. Der Schlüssel K wird mit ungerader Parität versehen. 128-Bit-KEKs können generiert werden, indem GKS zweimal mit C3L und C3R anstelle von C3 aufgerufen wird. Die Übergabe dieser Steuervektoren muß vom CFAP gesteuert werden.

$e*KEK1.C4(K)$  K ist ein zufallsmäßig generierter 64-Bit-Schlüssel, chiffriert unter KEK1 in Verbindung mit einem Steuervektor C4.

Der Steuervektor C4 wird immer benutzt. Der Schlüssel K wird mit ungerader Parität versehen. 128-Bit-KEKs können generiert werden, indem GKS zweimal mit C4L und C4R anstelle von C4 aufgerufen wird. Die Übergabe dieser Steuervektoren muß vom CFAP gesteuert werden.

Beschreibung: Die Anweisung GKS generiert eine mit ungerader Parität versehene 64-Bit-Zufallszahl mit zwei durch die Steuer-

vektoren C3 und C4 festgelegten Attributen. Ein 128-Bit-Schlüssel mit festgelegter Parität kann durch zweimalige Ausführung der Anweisung GKS generiert werden. Mit GKS können Schlüssel nur über einen CV-Kanal an CV-Systeme übertragen werden. Kompatibilitätsschlüssel werden mit GKS nicht erstellt. Diese werden mit KGEN generiert und müssen mit RFMK gesendet werden. Ein Kompatibilitätsschlüssel kann entweder über einen CV-Kanal (Steuervektoren werden mit übertragen) oder über einen CV=0-Kanal übertragen werden.

Die Anweisung GKS bietet hohe Sicherheit, da nur zwei Exemplare des generierten Schlüssels erstellt werden, die unter Schlüsseln, die zwei Zielsystemen gehören, chiffriert sind. Die Steuervektoren C3 und C4, die diesen beiden Exemplaren des generierten Schlüssels zugeordnet sind, können je nach Verwendungszweck der Schlüssel entweder identisch oder unterschiedlich sein.

Mit der Anweisung GKS können die in Fig. 32 aufgeführten Schlüssel generiert und verteilt werden.

Alle Schlüssel in CA-System werden unter einem 128-Bit-KEK dreifach chiffriert (EDE). Zur Übertragung von Schlüsseln an Knoten, die nur 64-Bit-KEKs unterstützen, wird ein verdoppelter 64-Bit-KEK benutzt. Wenn der Empfangsknoten 128-Bit-KEKs unterstützt, kann er einen Schlüssel durch Dreifach-Dechiffrierung (DED) rekonstruieren. Unterstützt der Empfangsknoten nur 64-Bit-KEKs, kann ein Schlüssel durch Einfach-Dechiffrierung rekonstruiert werden. Alle KEKs werden unter dem Hauptschlüssel in der Form  $e * KM.C1L(KEK) \parallel e * KM.C1R(IKEKR)$ , d.h. chiffriert in 128-Bit-Form, gespeichert. C1L ist eine linke Steuervektorhälfte und C1R eine rechte Steuervektorhälfte. Die Steuervektorhälften C1L und C1R unterscheiden sich im Schlüsselformfeld. Dadurch wird der Angriff auf Schlüssel mit doppelter Länge (mit ungleichen Hälften) mit Arbeitsfaktor  $2^{56}$  ausgeschaltet. Insider-Angriffe auf

einen 128-Bit-Schlüssel dürften in der Größenordnung von  $2^{56}$  liegen).

Fig. 33 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1L oder C1R ist ungültig
- 3. C2L oder C2R ist ungültig
- 4. C3 ist ungültig
- 5. C4 ist ungültig
- 6. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung: Für Modus = 0 (OP-OP) ist folgende Prüfung vorzuführen:

- 1. Prüfung von C3

-- CV-Art	=	"Daten/Vertraulichkeit" oder "Daten/MAC"
-- Anti-Variante(30)	=	'0'
-- Anti-Variante(38)	=	'1'
-- reserviert(48:63)	=	X'0'

- 2. Prüfung von C4

-- CV-Art	=	"Daten/Vertraulichkeit" oder "Daten/MAC"
-- Anti-Variante(30)	=	'0'
-- Anti-Variante(38)	=	'0'
-- reserviert(48:63)	=	X'0'

- 3. Prüfung von C3 & C4

- In Fig. 34 sind die gültigen Kombinationen der zu prüfenden Attribute von C3 und C4 aufgeführt. Alle nicht in der Tabelle aufgeführten Kombinationen sind ungültig und daher nicht zulässig.

Für Modus = 1 (OP-EX) ist folgende Prüfung vorzuführen:

- 1. Prüfung von C2L
  - CV-Art = "KEK/Sender"
  - GKS-Verwendungsbits (1) = 1
  - reserviert(48:63) = X'0'
- 2. Prüfung von C2R
  - CV-Art = "KEK/Sender"
  - GKS-Verwendungsbits (1) = 1
  - reserviert(48:63) = X'0'
- 3. Prüfung von C3
  - CV-Art = "Daten/Vertraulichkeit" oder "Daten/MAC" oder "Daten/XLT" oder "KEK/Sender" oder "KEK/-Empfänger" oder "PIN/PEK"
  - Anti-Variante(30) = '0'
  - Anti-Variante(38) = '1'
  - reserviert(48:63) = X'0'
- 4. Prüfung von C4
  - CV-Art = "Daten/Vertraulichkeit" oder "Daten/MAC" oder "Daten/XLT" oder "KEK/Sender" oder "KEK/-Empfänger" oder "PIN/PEK"

```

-- Anti-Variante(30)   =   '0'
-- Anti-Variante(38)   =   '1'
-- reserviert(48:63)   =   X'0'

```

- 5. Prüfung von C3 & C2L,C2R

```

-- Wenn CV-Art (C3)    =   "KEK/Sender" oder "KEK/Emp-
                           fänger" & Schlüsselform(C3) =
                           '10' oder '11', dann ist
                           Schlüsselform (C2L) = '10'
                           und Schlüsselform (C2R) =
                           '11'

```

- 6. Prüfung von C4 & C2L,C2R

```

-- Wenn CV-Art (C4)    =   "KEK/Sender" oder "KEK/Emp-
                           fänger" & Schlüsselform(C4) =
                           '10' oder '11', dann ist
                           Schlüsselform (C2L) = '10'
                           und Schlüsselform (C2R) =
                           '11'

```

- 7. Prüfung von C2L & C2R

```

-- In Fig. 35 sind die gültigen Kombinationen der zu prü-
   fenden Attribute von C2L und C2R aufgeführt. Alle
   nicht in der Tabelle aufgeführten Kombinationen sind
   ungültig und daher nicht zulässig.

```

- 8. Prüfung von C3 & C4

```

-- In Fig. 36 sind die gültigen Kombinationen der zu prü-
   fenden Attribute von C3 und C4 aufgeführt. Andere Kom-
   binationen sind nicht zulässig.

```

Für Modus = 2 (EX-EX) ist folgende Prüfung vorzuführen:

- 1. Prüfung von KEK1 und KEK2

-- KEK1L 11 KEK1R = KEK2L 11 (128-Bit-Schlüssel)

HINWEIS: Durch die Prüfung der KEKs auf Nichtübereinstimmung wird sichergestellt, daß keine bidirektionalen Schlüssel gesendet werden.

- 2. Prüfung von C1L

-- CV-Art = "KEK/Sender"

-- GKS-Verwendungsbits (2) = 1

-- reserviert(48:63) = X'0'

- 3. Prüfung von C1R

-- CV-Art = "KEK/Sender"

-- GKS-Verwendungsbits (2) = 1

-- reserviert(48:63) = X'0'

- 4. Prüfung von C2L

-- CV-Art = "KEK/Sender"

-- GKS-Verwendungsbits (2) = 1

-- reserviert(48:63) = X'0'

- 5. Prüfung von C2R

-- CV-Art = "KEK/Sender"

-- GKS-Verwendungsbits (1) = 1

-- reserviert(48:63) = X'0'

- 6. Prüfung von C3

-- CV-Art = "Daten/Vertraulichkeit" oder  
 "Daten/MAC" oder "Daten/XLT"  
 oder "KEK/Sender" oder "KEK/-  
 Empfänger" oder "PIN/PEK"  
 -- Anti-Variante(30) = '0'  
 -- Anti-Variante(38) = '1'  
 -- reserviert(48:63) = X'0'

- 7. Prüfung von C4

-- CV-Art = "Daten/Vertraulichkeit" oder  
 "Daten/MAC" oder "Daten/XLT"  
 oder "KEK/Sender" oder "KEK/-  
 Empfänger" oder "PIN/PEK"  
 -- Anti-Variante(30) = '0'  
 -- Anti-Variante(38) = '1'  
 -- reserviert(48:63) = X'0'

- 8. Prüfung von C1L & C1R

-- wie bei Fig. 35 beschrieben

- 9. Prüfung von C2L & C2R

-- wie bei Fig. 35 beschrieben

10. Prüfung von C3 & C4

-- wie bei Fig. 36 beschrieben

- 11. Prüfung von C3 & C2L,C2R

-- Wenn CV-Art (C3) = "KEK/Sender" oder "KEK/Emp-  
 fänger" & Schlüsselform(C3) =



'10' oder '11', dann ist  
 Schlüsselform (C2L) = '10'  
 und Schlüsselform (C2R) =  
 '11'

- 12. Prüfung von C4 & C2L,C2R

-- Wenn CV-Art (C4) = "KEK/Sender" oder "KEK/Empfänger" & Schlüsselform(C4) = '10' oder '11', dann ist Schlüsselform (C2L) = '10' und Schlüsselform (C2R) = '11'

Für Modus = 3 (OP-IM) ist folgende Prüfung vorzuführen:

- 1. Prüfung von C2L

-- CV-Art = "KEK/Empfänger"  
 -- GKS-Verwendungsbits (1) = 1  
 -- reserviert(48:63) = X'0'

- 2. Prüfung von C2R

-- CV-Art = "KEK/Empfänger"  
 -- GKS-Verwendungsbits (1) = 1  
 -- reserviert(48:63) = X'0'

- 3. Prüfung von C3

-- CV-Art = "Daten/Vertraulichkeit" oder "Daten/MAC" oder "Daten/XLT"  
 -- Anti-Variante(30) = '0'  
 -- Anti-Variante(38) = '1'  
 -- reserviert(48:63) = X'0'

- 4. Prüfung von C2L & C2R
  - wie bei Fig. 35 beschrieben.
- 5. Prüfung von C2L & C2R
  - wie bei Fig. 35 beschrieben.
- 6. Prüfung von C3 & C4
  - In Fig. 37 sind die gültigen Kombinationen der zu prüfenden Attribute von C3 und C4 aufgeführt. Nicht in der Tabelle aufgeführte Kombinationen sind ungültig und deshalb nicht zulässig.

Für Modus = 4 (IM-EX) ist folgende Prüfung vorzunehmen: (Dieser Modus wird für IBM SNA Mehrdomänenanwendungen benutzt.)

- 1. Prüfung von C1L
  - CV-Art = "KEK/Empfänger"
  - GKS-Verwendungsbits (1) = 1
  - reserviert(48:63) = X'0'
- 2. Prüfung von C1R
  - CV-Art = "KEK/Empfänger"
  - GKS-Verwendungsbits (1) = 1
  - reserviert(48:63) = X'0'
- 3. Prüfung von C2L
  - CV-Art = "KEK/Sender"
  - GKS-Verwendungsbits (1) = 1
  - reserviert(48:63) = X'0'

- 4. Prüfung von C2R

-- CV-Art = "KEK/Sender"  
 -- GKS-Verwendungsbits (1) = 1  
 -- reserviert(48:63) = X'0'

- 5. Prüfung von C3

-- CV-Art = "Daten/Vertraulichkeit" oder  
 "Daten/MAC" oder "Daten/XLT"  
 oder "KEK/Sender"  
 -- Anti-Variante(30) = '0'  
 -- Anti-Variante(38) = '1'  
 -- reserviert(48:63) = X'0'

- 6. Prüfung von C4

-- CV-Art = "Daten/Vertraulichkeit" oder  
 "Daten/MAC" oder "Daten/XLT"  
 oder "KEK/Sender" oder "KEK/-  
 Empfänger" oder "PIN/PEK"  
 -- Anti-Variante(30) = '0'  
 -- Anti-Variante(38) = '1'  
 -- reserviert(48:63) = X'0'

- 7. Prüfung von C1L & C1R

-- wie bei Fig. 35 beschrieben.

- 8. Prüfung von C2L & C2R

-- wie bei Fig. 35 beschrieben.

- 9. Prüfung von C3 & C4

- wie bei Fig. 36 beschrieben.
- 10. Prüfung von C3 & C2L,C2R
  - Wenn CV-Art (C3) = "KEK/Sender" oder "KEK/Empfänger" & Schlüsselform(C3) = '10' oder '11', dann ist Schlüsselform (C2L) = '10' und Schlüsselform (C2R) = '11'
- 11. Prüfung von C4 & C2L,C2R
  - Wenn CV-Art (C4) = "KEK/Sender" oder "KEK/Empfänger" & Schlüsselform(C4) = '10' oder '11', dann ist Schlüsselform (C2L) = '10' und Schlüsselform (C2R) = '11'

#### Umchiffrieren vom Hauptschlüssel (RFMK)

- Gleichung:  $\text{dist-mode}, e*KM.C1L(KEK1L), e*KM.C1R(KEK1R), e*KM.C2(K), C1L, C1R, C2, C3 \text{ --- } e*KEK1.C3(K)$
- Eingabe: dist-mode Kanalart zur Versendung eines Schlüssels.

Kanäle zur Versendung von Schlüsseln sind beispielsweise CV und CV=0.

- 0: CV = 0 (Kanal)
- 1: CV (Kanal)

$e*KM.C1L(KEK1L)$  KEK1L linker 64-Bit-Schlüsselchiffrierschlüssel, der eine

Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels KEK1 bildet, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C1R.

e*KM.C1L(KEK1L)	KEK1L	rechter 64-Bit-Schlüsselchiffrierschlüssel, der eine Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels KEK1 bildet, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C1R.
e*KM.C2(K)		K ist ein 64-Bit-Schlüssel, dreifach chiffriert unter KM in Verbindung mit einem Steuervektor C2.

C2 ist der Eingabe-Steuervektor zur Rekonstruktion des Schlüssels. K ist der zu exportierende Schlüssel. Ein 128-Bit-Schlüssel wird durch zweimalige Ausführung der Anweisung RFMK exportiert.

C1L,C1R	64-Bit-Steuervektoren, linker und rechter Steuervektor für 128-Bit-KEK1.
---------	--

C1L ist der Steuervektor für KEK1L und C1R der Steuervektor für KEK1R. Die tatsächliche Implementierung kann auch nur ein CV übergeben und die Schlüsselformbits im Steuervektor implizit von der CF setzen lassen. Bei der CA wurde entschieden, für KEKs den rechten und den linken Steuervektor separat zu übergeben. Diese beiden Steuervektorarten für die KEKs müssen vom CFAP generiert und verwaltet werden.

C2	64-Bit-Eingabesteuervektor für Schlüssel K.
C3	64-Bit-Ausgabesteuervektor für Schlüssel K.

- Ausgabe: e\*KEK1.C3(K) K ist der zu exportierende  
64-Bit-Schlüssel.

K kann mit C3 als Steuervektor oder mit  $C3 = 0$  als Steuervektor exportiert werden. Der Schlüssel K ist dreifach chiffriert unter einem 128-Bit-KEK1 mit Steuervektor C3. Bei  $KEK1L = KEK1R$  kann der Empfänger K durch einfache Dechiffrierung dieser Ausgabe rekonstruiert werden.

**Beschreibung:** Die Anweisung RFMK chiffriert einen unter dem Hauptschlüssel chiffrierten 64-Bit-Schlüssel K unter einen 128-Bit-KEK1 (den sog. Exportschlüssel) um. Ein 128-Bit-Schlüssel K (d.h. zwei 64-Bit-Schlüssel) kann durch zweimaliges Aufrufen der Anweisung RFMK exportiert werden. Ein Schlüssel, der an ein System mit Steuervektoren exportiert wird, ist unter KEK1.C3 chiffriert, d.h. der Steuervektor wird zusammen mit dem exportierten Schlüssel übertragen. Ein Schlüssel, der an ein System ohne Steuervektoren exportiert wird, ist unter KEK1.0 chiffriert, d.h. der Ausgabesteuervektor muß Null sein.

RFMK wird zum Exportieren von Schlüsseln verwendet, wenn drei oder mehr Schlüssel verteilt werden müssen (in diesem Fall können die erforderlichen Schlüssel nicht mit GKS generiert werden). RFMK wird außerdem zur Übertragung von Daten/Kompatibilitätsschlüsseln an Systeme ohne Steuervektoren verwendet. Wenn dist-mode = 0 ist, muß der angegebenen Steuervektor C3 Null sein. Datenschlüssel, KEKs, PIN-Chiffrierschlüssel, PIN-Generierungsschlüssel, Zwischen-ICVs, Schlüsselteile und Token können mit RFMK an ein CV-System übertragen werden. RFMK kann nicht zum Exportieren von durch GKS erstellten Schlüsseln über den CV=0-

Kanal verwendet werden. Diese Einschränkung soll die Trennung der Kanäle CV und CV=0 gewährleisten.

Fig. 38 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1L oder C1R ist ungültig
- 3. C2 ist ungültig
- 4. C3 ist ungültig
- 5. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1L (Exportschlüssel):

-- CV-Art = "KEK/Sender"  
 -- RFMK-Verwendungsbit = 1  
 -- reserviert(48:63) = X'0'

- 2. Prüfung von C1R (Exportschlüssel):

-- CV-Art = "KEK/Sender"  
 -- RFMK-Verwendungsbit = 1  
 -- reserviert(48:63) = X'0'

- 3. Prüfung von C2 (Exportierter Schlüssel):

-- CV-Art = "Daten/Kompatibilität" oder  
 "Daten/Vertraulichkeit" oder  
 "Daten/MAC" oder "Daten/XLT"  
 oder "KEK/Empfänger" oder  
 "KEK/Sender" oder "PIN/PEK"  
 oder "PIN/PGK" oder "Schlüs-

selteil" oder "Zwischen-ICV"  
oder "Token"

-- reserviert(48:63) = X'0'

- 4. Für dist-mode = 2 folgende Prüfung vonführen:

-- Verbindungssteuerung (C1L) = "01" oder "11"  
 -- Verbindungssteuerung (C1R) = "01" oder "11"  
 -- Exportkontrollbit (C2) = 1  
 -- CV-Art (C2) = CV-Art (C3)  
 -- Verwendungsbits(C2) = Verwendungsbits(C3)  
 -- Anti-Variantenbit 30(C3) = '0'  
 -- Anti-Variantenbit 38(C3) = '1'  
 -- reserviert (48:63) (C3) = X'0'

- 5. Für dist-mode = 0 folgende Prüfung vonführen:

-- Verbindungssteuerung (C1L) = "10" oder "11"  
 -- Verbindungssteuerung (C1R) = "10" oder "11"  
 -- Exportkontrollbit (C2) = 1  
 -- CV-Art (C2) = "Daten/Kompatibilität"  
 -- C3 = 0

- 6. Prüfung von C1L & C1R

-- wie bei Fig. 35 beschrieben.

- 7. Prüfung von C3 & C1L,C1R

-- Wenn CV-Art (C3) = "KEK/Sender" oder "KEK/Emp-  
 fänger" & Schlüsselform(C3) =  
 '10' oder '11', dann ist  
 Schlüsselform (C1L) = '10' &  
 Schlüsselform (C1R) = '11'.



Umchiffrieren unter Hauptschlüssel (RTMK)

- Gleichung:  $\text{dist-mode}, e*KM.C1L(KEK1L), e*KM.C1R(KEK1R),$   
 $e*KEK1.C32(K), C1L, C1R, C2, C3 \text{ --- } e*KM.C2(K)$
- Eingabe: `dist-mode` Kanalart zum Empfangen eines Schlüssels.

Kanäle zur Versendung von Schlüsseln sind beispielsweise CV und CV=0.

- 0: CV = 0 (Kanal)
- 1: CV (Kanal)

<code>e*KM.C1L(KEK1L)</code>	<code>KEK1L</code>	linker 64-Bit-Schlüsselchiffrierschlüssel, der eine Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels KEK1 bildet, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C1L.
<code>e*KM.C1R(KEK1R)</code>	<code>KEK1R</code>	rechter 64-Bit-Schlüsselchiffrierschlüssel, der eine Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels KEK1 bildet, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C1R. (KEK1 ist ein Eingabeschlüssel.)
<code>e*KEK1.C23K)</code>		K ist ein Schlüssel, der dreifach chiffriert unter dem Schlüsselchiffrierschlüssel KEK1 in Verbindung mit einem Steuervektor C3 von einem

anderen Knoten gesendet wurde.

Der Schlüssel kann einfach oder dreifach chiffriert übertragen werden. Er wird in beiden Fällen rekonstruiert. Der Schlüssel kann über einen CV-Kanal oder einen CV=0-Kanal gesendet werden. Bei Verwendung eines CV=0-Kanals muß dist-mode = 0 sein. Der empfangene Schlüssel kann ungerade Parität haben, muß aber nicht, da die Parität nicht von der CF geprüft werden muß. C3 wird auch als Eingabe-Steuervektor bezeichnet.

C1L, C1R	64-Bit-Steuervektoren, linker und rechter Steuervektor für 128-Bit-KEK1.
----------	--

C1L ist der Steuervektor für KEK1L und C1R der Steuervektor für KEK1R. Die tatsächliche Implementierung kann auch nur ein CV übergeben und die Schlüsselformbits im Steuervektor implizit von der CF setzen lassen. Bei der CA wurde entschieden, für KEKs den rechten und den linken Steuervektor separat zu übergeben. Diese beiden Steuervektorarten für die KEKs müssen vom CFAP generiert und verwaltet werden.

C2	64-Bit-Ausgabesteuervektor für Schlüssel K.
C3	64-Bit-Eingabesteuervektor für Schlüssel K.

-	Ausgabe:	$e * KM.C2(K)$	K ist der empfangene 64-Bit-Schlüssel, dreifach chiffriert unter KM in Verbindung mit einem Steuervektor C2.
---	----------	----------------	--

Der Schlüssel wird vor der Chiffrierung mit dem Hauptschlüssel mit ungerader Parität versehen.

Beschreibung: Die Anweisung RTMK chiffriert einen unter einem 128-Bit-Schlüsselchiffrierschlüssel KEK (dem sog. Importschlüssel) chiffrierten 64-Bit-Schlüssel K unter den Hauptschlüssel um. Ein 128-Bit-Schlüssel K (d.h. zwei 64-Bit-Schlüssel) kann durch zweimaliges Aufrufen der Anweisung RTMK exportiert werden.

Von einem Schlüssel, der von einem System mit Steuervektoren importiert wird, wird erwartet, daß er unter KEK1.C3 chiffriert ist, d.h., daß der Steuervektor zusammen mit dem importierten Schlüssel übertragen wird. Dies wird als Import über einen CV-Kanal bezeichnet. Andere Kanäle sind für den Import oder Export von Schlüsseln in CA-Systemen nicht verfügbar. In dist-mode muß der zur Übertragung des Schlüssels verwendete Kanal richtig angegeben werden. Bei dist-mode = 0 muß der Eingabe-Steuervektor Null sein. CV=0 kann nur zum Senden oder Empfangen von Daten/-Kompatibilitätsschlüsseln benutzt werden. Andere Schlüssel können in CA-Systemen nicht über diesen Kanal gesendet oder empfangen werden.

Der von einem anderen System importierte Schlüssel wird immer unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C2 chiffriert gespeichert. C2 muß der Anweisung immer übergeben werden. In CA-Systemen werden also alle Schlüssel mit Steuervektoren gespeichert.

Fig. 39 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1L oder C1R ist ungültig
- 3. C2 ist ungültig
- 4. C3 ist ungültig
- 5. fehlgeschlagene Operation (Fehler)

## # Steuervektorprüfung:

## - 1. Prüfung von C1L (Importschlüssel):

```
-- CV-Art                = "KEK/Empfänger"
-- RTMK-Verwendungsbit   = 1
-- reserviert(48:63)     = X'0'
```

## - 2. Prüfung von C1R (Importschlüssel):

```
-- CV-Art                = "KEK/Empfänger"
-- RTMK-Verwendungsbit   = 1
-- reserviert(48:63)     = X'0'
```

## - 3. Für dist-mode = 1 folgende Prüfung vonführen:

```
-- Verbindungssteuerung (C1L) = "01" oder "11"
-- Verbindungssteuerung (C1R) = "01" oder "11"
-- CV-Art                    = "Daten/Kompatibilität"
                                oder "Daten/Vertraulich-
                                keit" oder "Daten/MAC"
                                oder "Daten/XLT" oder
                                "KEK/Empfänger" oder
                                "KEK/Sender" oder "PIN/-
                                PEK" oder "PIN/PGK" oder
                                "Schlüsselteil" oder
                                "Zwischen-ICV" oder "To-
                                ken".
-- reserviert(48:63)        = X'0'
-- Wenn Exportkontrollbit (C3) = 0, dann ist Export-
                                kontrollbit (C2) = 0
-- CV-Art (C3)              = CV-Art (C2)
-- Verwendungsbits(C3)      = Verwendungsbits(C2)
-- Anti-Variantenbit 30(C3) = '0'
-- Anti-Variantenbit 38(C3) = '1'
```

- 4. Für dist-mode = 0 folgende Prüfung vonführen:
  - Verbindungssteuerung (C1L) = "10" oder "11"
  - Verbindungssteuerung (C1R) = "10" oder "11"
  - CV-Art (C2) = "Daten/Kompatibilität"
  - Anti-Variantenbit 30(C2) = '0'
  - Anti-Variantenbit 38(C2) = '1'
  - C3 = 0
- 5. Prüfung von C1L & C1R
  - wie bei Fig. 35 beschrieben.
- 6. Prüfung von C2 & C1L,C1R
  - Wenn CV-Art (C2) = "KEK/Sender" oder "KEK/Empfänger" & Schlüsselform(C2) = '10' oder '11', dann ist Schlüsselform (C1L) = '10' & Schlüsselform (C1R) = '11'.

#### Schlüssel generieren (KGEN)

- Gleichung:  $\text{Ausgabeart}, C1 \text{ --- } K, e * KM.C1(K)$
- Eingabe: Ausgabeart gibt das Ausgabeformat der Anweisung an.
  - 0: Ausgabe 64-Bit-Zufallszahl (Schlüssel) mit ungerader Paritätsanpassung
  - 1: Ausgabe 64-Bit-Zufallszahl ohne Paritätsanpassung
  - 2. Ausgabe 64-Bit-Zufallszahl (Schlüssel) mit ungerader Paritätsanpassung, chiffriert unter KM in Verbindung mit einem Steuervektor C1.

C1	64-Bit-Steuervektor für den generierten Schlüssel
- Ausgabe: K	64 Bit lange unchiffrierte Zufallszahl oder Schlüssel mit Paritätsanpassung.
e*KM.C1(K)	K ist der generierte 64-Bit-Schlüssel, dreifach chiffriert unter KM in Verbindung mit einem Steuervektor C1.

Beschreibung: Die Funktion KGEN erzeugt einen 64 Bit langen Schlüssel mit angepaßter Parität oder einen 64-Bit-Klartextschlüssel mit angepaßter ungerader Parität, der unter KM in Verbindung mit einem Steuervektor C1 chiffriert wird, oder eine 64-Bit-Zufallszahl ohne Paritätsanpassung. Die Ausgabeart legt die Art der erforderlichen Ausgabe fest. Mit KGEN können Schlüssel der Arten "Daten/Kompatibilität", "Daten/Vertraulichkeit", "Daten/MAC" und "Daten/XLT" sowie ANSI-Schlüssel generiert werden. Alle anderen Schlüsselarten müssen mit GKS generiert werden.

Fig. 40 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1 ist ungültig
- 3. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1 auf Ausgabeart = 2.

-- CV-Art = "Daten/Kompatibilität" oder  
 "Daten/Vertraulichkeit" oder  
 "Daten/MAC" oder "Daten/XLT"

oder "Daten/ANSI" oder "ANSI/KEK"

```
-- Anti-Variante(30)  =  '0'
-- Anti-Variante(38)  =  '1'
-- reserviert(48:63)  =  X'0'
```

### Chiffrieren unter Hauptschlüssel (EMK)

```
- Gleichung:      K, C1  ---- e*KM.C1(K)
- Eingabe:       K      64-Bit-Klartextschlüssel mit
                        angepaßter ungerader Parität,
                        der der Funktion übergeben
                        wird, oder 64-Bit-Token für
                        einen Datenschlüssel.
```

Das Token wird zusammen mit dem Datenschlüssel im CKDS gespeichert. Die Verwendung von Tokens für die Datenschlüssel wird bei der Anweisung RTNMK beschrieben.

```
      C1      64-Bit-Steuervektor für den
              übergebenen Schlüssel oder das
              Token.
- Ausgabe:  e*KM.C1(K)  K ist der übergebene 64-Bit-
                        Schlüssel, dreifach chiffriert
                        unter KM in Verbindung mit einem
                        Steuervektor C1.
```

K kann auch ein 64-Bit-Token sein, das mit dem Datenschlüssel zusammen im CKDS gespeichert wird.

Beschreibung: Die Funktion EMK erzeugt einen unter KM in Verbindung mit einem Steuervektor C1 chiffrierten 64-Bit-Schlüssel oder ein ebenso chiffriertes 64-Bit-Token.

Wenn es sich bei dem Schlüssel oder Token, der/das der Funktion übergeben wird, um die Art "Daten/Vertraulichkeit" oder "Token" handelt, wird er/es unter KM in Verbindung mit einem Steuervektor C1 chiffriert. Andernfalls kann die Funktion nur ausgeführt werden, wenn das System sich im "Extrasicher"-Modus befindet. In diesem Modus können mit EMK alle Klartextschlüssel mit jeder CV-Art chiffriert werden.

Fig. 41 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1 ist ungültig
- 3. Nicht im "Extrasicher"-Modus
- 4. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1:

```
-- CV-Art = "Daten/Kompatibilität" oder
           "Daten/Vertraulichkeit" oder
           "Daten/MAC" oder "Daten/XLT"
           oder "Daten/ANSI" oder "AN-
           SI/KEK" oder "KEK/Sender"
           oder "KEK/Empfänger" oder
           "Token" oder "Zwischen-ICV"
           oder "PIN/PGK" oder "PIN/PEK"

-- Anti-Variante(30) = '0'
-- Anti-Variante(38) = '1'
-- reserviert(48:63) = X'0'
-- Wenn (CV-Art () ("Daten/Kompatibilität" oder "Token"),
    dann ist supersecure-mode-flat = 1.
```



Schlüssel umwandeln (XLTKEY)

- Gleichung:       $e*KM.C1L(KEK1L)$ ,  $e*KM.C1R(KEK1R)$ ,  
                   $e*KM.C2L(KEK2L)$ ,  $e*KM.C2R(KEK2R)$ ,  
                   $e*KEK1.C3(K)$ , Modus, C1L, C1R, C2L, C2R,  
                  C3 ----  $e*KEK2.C3(K)$
- Eingabe:
 

$e*KM.C1L(KEK1L)$	KEK1L ist ein linker 64-Bit-Schlüsselchiffrierschlüssel, der Bestandteil eines 128-Bit-Schlüsselchiffrierschlüssels KEK1 ist, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C1L (Import-KEK).
$e*KM.C1R(KEK1R)$	KEK1R ist ein rechter 64-Bit-Schlüsselchiffrierschlüssel, der Bestandteil eines 128-Bit-Schlüsselchiffrierschlüssels KEK1 ist, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C1R (Import-KEK).
$e*KM.C2L(KEK2L)$	KEK2L ist ein linker 64-Bit-Schlüsselchiffrierschlüssel, der Bestandteil eines 128-Bit-Schlüsselchiffrierschlüssels KEK2 ist, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C2L (Import-KEK).
$e*KM.C2R(KEK2R)$	KEK2R ist ein rechter 64-Bit-Schlüsselchiffrierschlüssel, der Bestandteil eines 128-Bit-Schlüsselchiffrierschlüssels KEK2 ist, dreifach chiffriert unter dem

131

Hauptschlüssel KM in Verbindung  
mit einem Steuervektor C2R  
(Import-KEK).

e\*KEK1.C3(K)

K ist ein 64-Bit-Schlüssel,  
dreifach chiffriert unter KM in  
Verbindung mit einem Steuervektor  
C3. K ist der umzuwandelnde  
Schlüssel. Ein 128-Bit-Schlüssel  
wird durch zweimaliges Aufrufen  
der Anweisung XLTKEY umgewandelt.

Modus

0/1

- 0: C3 = von Null verschiedener Steuervektor
- 1: C3 = 0

C1L,C1R

64-Bit-Steuervektoren, linker und  
rechter Steuervektor für 128-Bit-  
KEK1.

C1L ist der Steuervektor für KEK1L und C1R der Steuervektor für  
KEK1R. Die tatsächliche Implementierung kann auch nur ein CV  
übergeben und die Schlüsselformbits im Steuervektor implizit von  
der CF setzen lassen. Bei der CA wurde entschieden, für KEKs den  
rechten und den linken Steuervektor separat zu übergeben. Diese  
beiden Steuervektorarten für die KEKs müssen vom CFAP generiert  
und verwaltet werden.

C2L,C2R

64-Bit-Steuervektoren, linker und  
rechter Steuervektor für 128-Bit-  
KEK2.

C2L ist der Steuervektor für KEK2L und C2R der Steuervektor für  
KEK2R.

C3

64-Bit-Steuervektor für den  
Schlüssel K.

Der gleiche Steuervektor wird auch für die Ausgabe des Schlüssels unter KEK2 verwendet.

- Ausgabe:  $e * KEK2.C3(K)$  umgewandelter 64-Bit-Schlüssel.

Beschreibung: Die Anweisung XLTKEY chiffriert einen unter dem 128-Bit-Schlüsselchiffrierschlüssel KEK1 (dem Importschlüssel) chiffrierten 64-Bit-Schlüssel um unter den 128-Bit-Schlüsselchiffrierschlüssel KEK2 (den Exportschlüssel). Lokal werden alle KEKs in 128-Bit-Form unter dem Hauptschlüssel in Verbindung mit einem Steuervektor gespeichert. Die Hälften KEKL und KEKR des 128-Bit-KEK werden separat unter KM chiffriert, so daß zwei 64-Bit-Schlüsselteile entstehen, die lokal zu 128 Bit verkettet gespeichert werden. Um den 128-Bit-KEK zu rekonstruieren, sind zwei Dechiffrierungen der Art DED erforderlich. Ein 128-Bit-Schlüssel K (d.h. zwei 64-Bit-Schlüssel) kann durch zweimaligen Aufruf der Anweisung XLTKEY umgewandelt werden.

Der Modusparameter gibt die Schlüsselart des umzuwandelnden Schlüssels an. Der Schlüssel muß mit diesem Modusparameter rekonstruiert werden: Wenn Modus = 1 ist, muß zur Rekonstruktion und auch zur Umwandlung des Schlüssels C3 = 0 verwendet werden. Mischen und Abgleichen von CV = 0 mit CV ist nicht erlaubt.

Fig. 42 ist ein Blockdiagramm dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. C1L oder C1R ist ungültig
- 3. C2L oder C2R ist ungültig
- 4. C3 ist ungültig
- 5. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1L (Importschlüssel):
  - CV-Art = "KEK/Empfänger"
  - XLTin-Verwendungsbit = 1
  - reserviert (48:63) = X'0'
- 2. Prüfung von C1R (Importschlüssel):
  - CV-Art = "KEK/Empfänger"
  - XLTin-Verwendungsbit = 1
  - reserviert (48:63) = X'0'
- 3. Prüfung von C2L (Exportschlüssel):
  - CV-Art = "KEK/Sender"
  - XLTout-Verwendungsbit = 1
  - reserviert (48:63) = X'0'
- 4. Prüfung von C2R (Exportschlüssel):
  - CV-Art = "KEK/Sender"
  - XLTout-Verwendungsbit = 1
  - reserviert (48:63) = X'0'
- 5. Prüfung von C1L & C2L:
    - wie bei Fig. 43 beschrieben.
  - 6. Prüfung von C1R & C2R:
    - wie bei Fig. 43 beschrieben.
  - Prüfung von C1L, C1R, C2L, C2R:
    - Für Modus = 0:  
 (Verbindungssteuerung(C1L) = (Verbindungssteuerung(C1R) = '01' oder

```

(Verbindungssteuerung(C1L) = (Verbindungssteue-
    rung(C1R) = '11'
-- Für Modus = 0:
(Verbindungssteuerung(C2L) = (Verbindungssteue-
    rung(C2R) = '01' oder
(Verbindungssteuerung(C2L) = (Verbindungssteue-
    rung(C2R) = '11'
-- Für Modus = 1:
(Verbindungssteuerung(C1L) = (Verbindungssteue-
    rung(C1R) = '10' oder
(Verbindungssteuerung(C1L) = (Verbindungssteue-
    rung(C1R) = '11'
-- Für Modus = 1:
(Verbindungssteuerung(C2L) = (Verbindungssteue-
    rung(C2R) = '10' oder
(Verbindungssteuerung(C2L) = (Verbindungssteue-
    rung(C2R) = '11'

```

#### 8. Prüfung von C3 & C1L,C1R:

```

-- Wenn CV-Art (C3)      = "KEK/Sender" oder "KEK/Emp-
                           fänger" & Schlüsselform (C3)
                           = '10' oder '11', dann ist
Schlüsselform (C1L) = '10' und Schlüsselform (C1R)
                           = '11'.

```

#### 8. Prüfung von C3 & C2L,C2R:

```

-- Wenn CV-Art (C3)      = "KEK/Sender" oder "KEK/Emp-
                           fänger" & Schlüsselform (C3)
                           = '10' oder '11', dann ist
Schlüsselform (C2L) = '10' und Schlüsselform (C2R)
                           = '11'.

```

Umchiffrieren in neuen Hauptschlüssel (RTNMK)

- Gleichung:

Schlüsselmodus	Modus, $e*KMC.C1(K)$ , C1 ---- $e*KMN.C1(K)$
Token-Modus	Modus, $(Token + e*KMC.C1(K))$ , $e*KMC.C2(Token)$ , C1, C2 ---- (To- ken + $e*KMN.C1(K)$ , $e*KMN.C2(Token)$

- Eingabe:            Modus            bezeichnet den RTNMK-Modus wie folgt:

-- 0: Schlüsselmodus  
-- 1: Token-Modus

Im Schlüsselmodus werden Schlüssel in neue Hauptschlüssel umgewandelt. Im Token-Modus werden Schlüssel und Token in einen neuen Hauptschlüssel umgewandelt, der im DKDS in einer speziellen Form gespeichert wird. Das CFAP generiert eine Zufallszahl ordnet sie als Token zu. Der chiffrierte Datenschlüssel wird durch EXKLUSIV ODER mit diesem Token verknüpft, und das Ergebnis wird auf dem DKDS gespeichert. Das Token selbst ist geheim. Es wird mit EMK ( $e*KM.C2(Token)$ ) chiffriert und ebenfalls auf dem DKDS gespeichert. Wenn ein Hauptschlüssel geändert wird, müssen Token und Schlüssel mittels RTNMK als autarker Operation umchiffriert werden.

$e*KMC.C1(K)$	K ist ein 64-Bit-Schlüssel, dreifach chiffriert unter dem aktuellen Hauptschlüssel in Verbindung mit einem Steuervektor C1.
C1	ist ein Steuervektor für den 64-Bit-Schlüssel K.
$(Token + e*KMC.C1(K))$	ist eine 64 Bit lange Zeichenkette speziell für die Speicherung auf

dem DKDS ('+' steht für eine EXKLUSIV-ODER-Verknüpfung).

Das CFAP generiert diese Zeichenkette und speichert sie auf dem DKDS. Diese Eingabe ist für Modus = 1 gültig.

**e\*KMC.C2(Token)** ist ein 64-Bit-Token, chiffriert unter dem aktuellen Hauptschlüssel in Verbindung mit einem Steuervektor C2.

Dieses Token wird mit der Anweisung EMK generiert. Die Eingabe ist für Modus = 1 gültig.

**C2** ist ein Steuervektor für das 64-Bit-Token. Diese Eingabe ist für Modus = 1 gültig.

- **Ausgabe:**

**e\*KMN.C1(K) K** ist ein 64-Bit-Schlüssel, dreifach chiffriert unter dem neuen Hauptschlüssel in Verbindung mit einem Steuervektor C1.

**(Token + e\*KMN.C1(K))** ist eine spezielle unter dem neuen Hauptschlüssel generierte Zeichenkette, die auf dem DKDS gespeichert werden kann. Diese Ausgabe ist nur für Modus = 1 gültig.

**e\*KMN.C2(Token)** ein Token, das unter dem neuen Hauptschlüssel in Verbindung mit einem Steuervektor C2 neu chiffriert wurde. Diese Ausgabe ist nur für Modus = 1 gültig.

**Beschreibung:** Die Anweisung RTNMK bewirkt, daß ein chiffrierter 64-Bit-Schlüssel unter dem aktuellen Hauptschlüssel dechiffriert

und anschließend unter einem neuen Hauptschlüssel wieder chiffriert wird.

Mit der Anweisung RTNMK werden alle Schlüssel im System neu chiffriert, wenn ein Hauptschlüssel geändert wird. Diese Anweisung behält den aktuellen Hauptschlüssel und den neuen Hauptschlüssel in den entsprechenden Registern in der Chiffriervorrichtung. Um die Anweisung RTNMK auszuführen, muß eine Markierung für den neuen Hauptschlüssel (NMK-Markierung) vorhanden sein, die in der Chiffriervorrichtung gesetzt wird. Beim Laden des neuen Hauptschlüssels in die Chiffriervorrichtung wird diese Markierung gesetzt, und am Übergangspunkt wird die NMK-Markierung durch die Anweisung SMK zurückgesetzt. In der Chiffriervorrichtung muß es Vorrichtungen für den neuen und den aktuellen Hauptschlüssel geben; außerdem kann eine Speicherstelle für den alten Hauptschlüssel vorhanden sein, an der der vor dem aktuellen Hauptschlüssel zuletzt gültige alte Hauptschlüssel gespeichert wird. Jedem Hauptschlüssel muß in der CF eine Markierung zugeordnet sein, und die Markierung für den aktuellen Hauptschlüssel muß gesetzt sein, damit die RTNMK-Operation ausgeführt werden kann. Die Hauptschlüsselmarkierungen werden am Übergangspunkt durch die Anweisung SMK gesetzt.

In manchen Systemen gibt es vielleicht Anwendungen, die am Übergangspunkt offline sind. Einige Schlüssel sind deshalb möglicherweise noch unter dem alten Hauptschlüssel chiffriert. Wenn die Anwendung nach dem Übergangspunkt wieder online ist, können solche Schlüssel mit der Anweisung "Umchiffrieren in den aktuellen Hauptschlüssel" (RTCMK) vom alten Hauptschlüssel unter den aktuellen Hauptschlüssel chiffriert werden.

Wenn die CMK- und NMK-Markierungen nicht gesetzt sind, wird die Operation abgebrochen.

Die Anweisung RTNMK kann auch zur Umchiffrierung eines Tokens und eines dem Token zugeordneten Sonderwertes in einer autarken



Operation verwendet werden. Der Sonderwert wird durch EXKLUSIV-ODER-Verknüpfung eines Tokens mit dem chiffrierten Datenschlüssel berechnet. Die Datenschlüssel werden auf dem DKDS als solche Sonderwerte gespeichert, und der chiffrierte Token-Wert wird ebenfalls als geheimer Wert gespeichert. Der chiffrierte Token-Wert kann nur intern in den Anweisungen RTNMK oder RTCMK innerhalb der CF dechiffriert werden. Eine andere Möglichkeit, das Token zu dechiffrieren, gibt es in der CA nicht. Der Token-Wert wird mittels EMK chiffriert, und dem Token wird ein spezieller Steuervektor zugeordnet. Bei den Anweisungen EMK und RTNMK wird eine Steuervektorprüfung durchgeführt, damit die Sicherheit des Tokens im System gewährleistet ist.

Fig. 44 und Fig. 45 sind Blockdiagramme dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. NMK-Markierung nicht gesetzt, unzulässige Folge
- 3. CMK-Markierung nicht gesetzt, unzulässige Folge
- 4. C1 ist ungültig
- 5. C2 ist ungültig
- 6. fehlgeschlagene Operation (Fehler).

# Steuervektorprüfung:

- 1. Prüfung auf C1, falls Modus = 0 ist.
  - reserviert (48:63) = X'0'
- 2. Prüfung auf C2, falls Modus = 1 ist.
  - CV-Art = "Token"
  - reserviert (48:63) = X'0'

Umchiffrieren in aktuellen Hauptschlüssel (RTCMK)

- Gleichung:

Schlüsselmodus	Modus, e*KMO.C1(K), C1 ---- e*KMC.C1(K)
Token-Modus	Modus, (Token + e*KMO.C1(K)), e*KMO.C2(Token), C1, C2 ---- (Token + e*KMC.C1(K), e*KMC.C2(Token))

- Eingabe:            Modus    bezeichnet den RTCMK-Modus wie folgt:

-- 0: Schlüsselmodus  
-- 1: Token-Modus

e*KMO.C1(K)	K ist ein 64-Bit-Schlüssel, dreifach chiffriert unter dem alten Hauptschlüssel in Verbindung mit einem Steuervektor C1.
C1	ist ein Steuervektor für den 64-Bit-Schlüssel K.
(Token + e*KMO.C1(K))	ist eine 64 Bit lange Zeichenkette speziell für die Speicherung auf dem DKDS ('+' steht für eine EX-KLUSIV-ODER-Verknüpfung).

☞ Das CFAP generiert diese Zeichenkette und speichert sie auf dem DKDS. Diese Eingabe ist für Modus = 1 gültig.

e*KMO.C2(Token)	ist ein 64-Bit-Token, chiffriert unter dem alten Hauptschlüssel in Verbindung mit einem Steuervektor C2.
-----------------	--

Dieses Token wird mit der Anweisung EMK generiert. Die Eingabe ist für Modus = 1 gültig.

C2

ist ein Steuervektor für das 64-Bit-Token. Diese Eingabe ist für Modus = 1 gültig.

- Ausgabe:

e\*KMC.C1(K) K

ist ein 64-Bit-Schlüssel, dreifach chiffriert unter dem aktuellen Hauptschlüssel in Verbindung mit einem Steuervektor C1.

(Token + e\*KMC.C1(K))

ist eine spezielle unter dem neuen Hauptschlüssel generierte Zeichenkette, die auf dem DKDS gespeichert werden kann. Diese Ausgabe ist nur für Modus = 1 gültig.

e\*KMC.C2(Token)

ein Token, das unter dem aktuellen Hauptschlüssel in Verbindung mit einem Steuervektor C2 neu chiffriert wurde. Diese Ausgabe ist nur für Modus = 1 gültig.

Beschreibung: Die Anweisung RTCMK bewirkt, daß ein chiffrierter 64-Bit-Schlüssel unter dem alten Hauptschlüssel dechiffriert und anschließend unter einem aktuellen Hauptschlüssel wieder chiffriert wird.

☛

Mit der Anweisung RTCMK werden die Schlüssel im System neu chiffriert, wenn sie unter dem alten Hauptschlüssel chiffriert sind und der Hauptschlüssel seither geändert wurde. Dies bedeutet, daß die NMK-Markierung nicht gesetzt ist und die Anweisung SMK bereits ausgeführt wurde. Der aktuelle Hauptschlüssel ist der neue Hauptschlüssel für die Schlüssel, die nicht mit RTNMK umchiffriert wurden. Um die Anweisung RTCMK auszuführen, muß die CMK-Markierung und die OMK-Markierung gesetzt werden, und die NMK-Markierung muß in der CF zurückgesetzt werden. Bei dieser Anweisung bleiben der aktuelle Hauptschlüssel und der neue

Hauptschlüssel in den entsprechenden Registern in der Chiffrier-  
vorrichtung gespeichert.

Das CFAP im Chiffriersubsystem muß über die Verwendung der An-  
weisungen RTNMK und RTCMK sowie über die Hauptschlüsselebene  
informiert sein. Nur ein alter Hauptschlüssel kann wahlweise in  
der CF verwaltet werden, um das Problem der alten Hauptschlüssel  
zu lösen, und das CFAP muß deshalb immer wissen, welche Schlüs-  
sel unter dem alten bzw. unter dem aktuellen Hauptschlüssel  
chiffriert sind.

RTCMK ist eine nicht obligatorische Anweisung, die nur dann im-  
plementiert werden kann, wenn sie nach dem Übergangspunkt des  
neuen Hauptschlüssels ausgeführt werden soll.

Wenn die OMK- und CMK-Markierungen nicht gesetzt sind und die  
NMK-Markierung nicht zurückgesetzt ist, muß die Operation abge-  
brochen werden.

Die Anweisung RTCMK kann auch zur Umchiffrierung eines Tokens  
und eines dem Token zugeordneten Sonderwertes in einer autarken  
Operation verwendet werden. Der Sonderwert wird durch EXKLUSIV-  
ODER-Verknüpfung eines Tokens mit dem chiffrierten Datenschlüs-  
sel berechnet. Die Datenschlüssel werden auf dem DKDS als solche  
Sonderwerte gespeichert, und der chiffrierte Token-Wert wird  
ebenfalls als geheimer Wert gespeichert. Der chiffrierte Token-  
Wert kann nur intern in den Anweisungen RTNMK oder RTCMK inner-  
halb der CF dechiffriert werden. Eine andere Möglichkeit, das  
Token zu dechiffrieren, gibt es in der CA nicht. Der Token-Wert  
wird mittels EMK chiffriert, und dem Token wird ein spezieller  
Steuervektor zugeordnet. Bei den Anweisungen EMK und RTCMK wird  
eine Steuervektorprüfung durchgeführt, damit die Sicherheit des  
Tokens im System gewährleistet ist.

Fig. 46 und Fig. 47 sind Blockdiagramme dieser Anweisung.

# CC:

- 1. erfolgreiche Operation
- 2. OMK-Markierung nicht gesetzt, unzulässige Folge
- 3. CMK-Markierung nicht gesetzt, unzulässige Folge
- 4. NMK-Markierung nicht gesetzt, unzulässige Folge
- 5. C1 ist ungültig
- 6. C2 ist ungültig
- 7. fehlgeschlagene Operation (Fehler).

# Steuervektorprüfung:

- 1. Prüfung auf C1, falls Modus = 0 ist.
  - reserviert (48:63) = X'0'
- 2. Prüfung auf C2, falls Modus = 1 ist.
  - CV-Art = "Token"
  - reserviert (48:63) = X'0'

#### Hauptschlüssel setzen (SMK)

- Gleichung: ()
- Eingabe: keine
- Ausgabe: keine

Beschreibung: Die Anweisung SMK bewirkt folgendes:

- 1. (alter Hauptschlüssel = aktueller Hauptschlüssel)
- 2. (OMK-Markierung = 1)
- 3. aktueller Hauptschlüssel = neuer Hauptschlüssel
- 4. CMK-Markierung = 1
- 5. NMK-Markierung = 0.

Diese Anweisung darf nur im "Extrasicher"-Modus ausgeführt werden, d.h., die physische Schlüsselposition muß die "Extrasicher"-Position sein, und in der Chiffriervorrichtung muß die "Extrasicher"-Markierung gesetzt sein, damit diese Anweisung ausgeführt werden kann. Die linke und die rechte Hälfte des Hauptschlüssels werden im Register für den neuen Hauptschlüssel auf Nichtübereinstimmung geprüft; bei Übereinstimmung der beiden Schlüsselhälften muß die Operation abgebrochen werden.

Nach Ausführung von SMK in der CF verwendet diese für alle Chiffrieroperationen nur noch den neuen Hauptschlüssel. Nach Ausführung dieser Anweisung ist der Übergangspunkt wirksam. Enthält das System weitere Schlüssel, die unter dem alten Hauptschlüssel chiffriert sind, ist die Anweisung RTCMK die einzige Möglichkeit, die Schlüssel umzuchiffrieren.

Wenn die NMK-Markierung in der CF nicht gesetzt ist, muß die Operation abgebrochen werden.

# CC:

- 1. erfolgreiche Operation
- 2. NMK-Markierung nicht gesetzt
- 3. linke und rechte Hälfte identisch
- 4. Extrasicher-Markierung nicht gesetzt, ungültige Folge
- 5. fehlgeschlagene Operation (Fehler).

\*\* MDCOP \*\*

Beschreibung: Eine Anweisung nach dem Stand der Technik, die einen Änderungserkennungscode (MDC) berechnet.

#### Speicherinhalt der Chiffriervorrichtung löschen (CLRCF)

- Gleichung:       ()
- Eingabe:       keine

- Ausgabe: keine

Beschreibung: Die Register der Chiffriervorrichtung (CF), die Markierungen und der gesamte lokale Speicherinhalt werden gelöscht. Mit dieser Funktion können die Hauptschlüssel und Markierungen gelöscht werden, wenn der Hauptschlüssel in Gefahr ist. Wenn die Hauptschlüsselmarkierungen zurückgesetzt werden, kann ein Eindringling die Schlüssel nicht mit Hilfe von RTNMK oder RTCMK vom gefährdeten Schlüssel unter seinen eigenen Hauptschlüssel umchiffrieren.

Diese Funktion kann unter der Kontrolle eines physischen Schlüssels oder als privilegierte Anweisung implementiert werden, je nachdem, wie stark das System vor einer Funktionsunfähigkeit durch unberechtigte Benutzer geschützt werden soll.

# CC:

- 1. erfolgreiche Operation
- 2. fehlgeschlagene Operation (Fehler).

#### Schlüsselteileregister löschen (CLRKPR)

- Gleichung: ()  
 - Eingabe: keine  
 - Ausgabe: keine

Beschreibung: Die Anweisung CLRKPR löscht das Schlüsselteileregister in der Chiffriervorrichtung (CF). Mit dieser Anweisung können fehlerhaft eingegebene oder gefährdete Werte im Schlüsselteileregister gelöscht werden. CLRKPR ermöglicht das Löschen des Schlüsselteileregisters ohne die ganze CF zu stören; das Schlüsselteileregister wird aber auch mit der Anweisung CLRCF gelöscht. Es bleibt dem Betreiber überlassen, wie die Anweisungen CLRCF und CLRKPR implementiert werden; es kann beispielsweise eine einzige Anweisung geben, mit der ein CF-Register selektiert wird.

tiv zurückgesetzt werden kann. In einer Implementierung kann zum Beispiel ein physischer Schlüssel verwendet oder die Funktion als privilegierte Anweisung ausgeführt werden, je nachdem, wie stark das System vor einer Funktionsunfähigkeit durch unberechtigte Benutzer geschützt werden soll.

# CC:

- 1. erfolgreiche Operation
- 2. fehlgeschlagene Operation (Fehler)

#### NMK-Register löschen (CLRNMK)

- Gleichung:       ()
- Eingabe:       keine
- Ausgabe:       keine

Beschreibung: Die Anweisung CLRNMK löscht das NMK-Register in der Chiffriervorrichtung (CF). Mit dieser Anweisung können fehlerhaft eingegebene oder gefährdete Werte im NKM-Register gelöscht werden. Die Anweisung CLRCF ist zu umfassend, deshalb steht zum Löschen eines Registers für den neuen Hauptschlüssel die Anweisung CLRNMK zur Verfügung. Es bleibt dem Betreiber überlassen, wie Anweisungen wie CLFCF, CLRKPR und CLRNMK implementiert werden; es kann beispielsweise eine einzige Anweisung geben, mit der ein CF-Register selektiv zurückgesetzt werden kann. In einer Implementierung kann zum Beispiel ein physischer Schlüssel verwendet oder die Funktion als privilegierte Anweisung ausgeführt werden, je nachdem, wie stark das System vor einer Funktionsunfähigkeit durch unberechtigte Benutzer geschützt werden soll.

# CC:

- 1. erfolgreiche Operation
- 2. fehlgeschlagene Operation (Fehler)



## CV-Berechtigung reduzieren (LCVA)

- ```

- Gleichung:      e*KM.C1(K), C1, C2 ---- e*KM.C2(K)
- Eingabe:   e*KM.C1(K)      K ist ein 64-Bit-Schlüssel,
                                dreifach chiffriert unter dem
                                Hauptschlüssel in Verbindung mit
                                einem Steuervektor C1.
                                C1      64-Bit-Eingabesteuervektor
                                C2      64-Bit-Ausgabesteuervektor
- Ausgabe:   e*KM.C2(K)      K ist ein 64-Bit-Schlüssel,
                                dreifach chiffriert unter KM in
                                Verbindung mit einem 64-Bit-
                                Steuervektor C2.

```

**Beschreibung:** LCVA ist eine Steuervektoranweisung zur Reduzierung der Berechtigung im Exportkontrollfeld des Steuervektors.

# CC:

- 1. erfolgreiche Operation
- 2. C1 oder C2 ist ungültig
- 3. fehlgeschlagene Operation (Fehler).

## # Steuervektorprüfung:

- ```

-      1.      Prüfung auf C1 und C2

--      CV-Art (C1)                =      CV-Art (C2)
--      reserviert (48:63) (C1)    =      reserviert (48:63) (C2)
                                         =      X'0'
--      Verwendungsbits für        =      Verwendungsbits für
      KEKs (C1)                    KEKs (C2)
--      Verwendungsbits für        =      Verwendungsbits für
      PIN-Schlüssel (C1)          PIN-Schlüssel (C2)
--      Exportkontrollbits:
      Exportkontrollbit 1 (C2) =      1 (keine weitere RFMK)

```

Ersten Hauptschlüsselteil laden (LFMKP)

- Gleichung:       ()
- Eingabe:       keine
- Ausgabe:       keine

Beschreibung: Diese Anweisung lädt den Inhalt des KP-Registers in das NMK-Register - Markierung(NMK Reg) = "teilweise voll" und Markierung(KP Reg) = "leer". Der im Schlüsselteileregister gespeicherte erste Teil des Hauptschlüssels wird in das Register für den neuen Hauptschlüssel (NMK-Register) geladen. Außerdem wird die Markierung des NMK-Registers auf von "leer" auf "teilweise voll" gesetzt, um anzuzeigen, daß das NMK-Register nicht ganz voll ist, und der Inhalt des Schlüsselteileregisters wird von "voll" auf "leer" gesetzt, um anzuzeigen, daß das Schlüsselteileregister jetzt leer ist. Diese Operation wird nur ausgeführt, wenn das Schlüsselteileregister "voll" und das NMK-Register "leer" ist.

HINWEIS: Es wird davon ausgegangen, daß vor der Ausführung dieser Anweisung ein erster Hauptschlüsselteil über eine Eingabevorrichtung (z.B. ein Tastenfeld oder eine Tastatur) mit Triggertaste (z.B. Eingabetaste) in das Schlüsselteileregister eingegeben wurde. (Die Triggertaste dient gewöhnlich zum Starten der Aktion, in der der Wert des über das Tastenfeld oder die Tastatur eingegebenen Schlüssels in das Schlüsselteileregister geladen wird.)

Die folgende Zusatzfunktion ist nicht obligatorisch. Sie kann bei Systemen mit physischen Schlüsseln implementiert werden.

Damit diese Anweisung ausgeführt wird, muß außer den oben genannten Bedingungen für Markierungen und Register die Schlüssel-schalterstellung festgestellt werden und der Schlüsselschalter sich in der richtigen Stellung befinden (z.B. Aktivierung des ersten Hauptschlüsselteils).

# CC:

- 1. erfolgreiche Operation
- 2. CKP1 ist ungültig
- 3. fehlgeschlagene Operation (Fehler).

# Steuervektorprüfung: keine

Schlüsselteile kombinieren (CMKP)

- Gleichung:  $\text{Modus} \text{ === } \text{Inhalt des NMK} = (\text{Inhalt des NMK-Registers}) \text{ XOR } (\text{Inhalt des KP-Registers}), \text{ Markierung(KP-Register)} = \text{"leer"} \text{ und Markierung(NMK)} = \text{"teilweise voll"} \text{ bei Modus} = 0, \text{ oder Markierung(NMK)} = \text{"voll"} \text{ bei Modus} = 1.$
- Eingabe: keine
- Ausgabe: keine

"Modus" ist die Eingabe für die Anweisung und gibt an, ob es sich um den letzten Hauptschlüsselteil handelt.

- Wenn Modus = 0 ist, ist der zu kombinierende Schlüsselteil nicht der letzte, und es wird erwartet, daß noch weitere Schlüsselteile erwartet werden, um den vollständigen Hauptschlüssel zu bilden.
- Wenn Modus = 1 ist, ist der zu kombinierende Schlüsselteil der letzte, und nach Ausführung dieser Anweisung wird im NMK-Register der vollständige Hauptschlüssel gebildet.

Beschreibung: Bei dieser Anweisung wird der im Schlüsselteilregister gespeicherte j-te Hauptschlüsselteil KPj mit einem oder mehreren anderen im NMK-Register gespeicherten Schlüsselteilen kombiniert und das Ergebnis im NMK-Register gespeichert.

Die Anweisung ändert ferner folgende Markierungen:

- Die Markierung des KP-Registers wird von "voll" auf "leer" gesetzt.
- Die Markierung des NMK-Registers wird bei Modus = 1 auf "voll" und bei Modus = 0 auf "teilweise voll" gesetzt.

Diese Anweisung wird nur ausgeführt, wenn das Schlüsselteileregister "voll" und das NMK-Register "teilweise voll" ist.

HINWEIS: Es wird davon ausgegangen, daß vor der Ausführung dieser Anweisung der Hauptschlüsselteil KPj über eine Eingabevorrichtung (z.B. ein Tastenfeld oder eine Tastatur) mit Triggertaste (z.B. Eingabetaste) in das Schlüsselteileregister eingegeben wurde. (Die Triggertaste dient gewöhnlich zum Starten der Aktion, in der der Wert des über das Tastenfeld oder die Tastatur eingegebenen Schlüssels in das Schlüsselteileregister geladen wird.)

Die Verwendung dieser Anweisung und der Anweisung "Ersten Hauptschlüsselteil laden" bei der Installation eines mehrteiligen Hauptschlüssels wird im Abschnitt über die Schlüsselverwaltung bei den Schlüsselinstallationsprozeduren beschrieben. (Ein Hauptschlüssel KM kann beispielsweise aus n Teilen KM1, KM2, ..., KMn bestehen, wobei  $KM = KM1 \text{ XOR } KM2 \dots \text{ XOR } KMn$  ist.)

Die folgende Zusatzfunktion ist nicht obligatorisch. Sie kann bei Systemen mit physischen Schlüsseln implementiert werden.

Damit diese Anweisung ausgeführt wird, muß außer den oben genannten Bedingungen für Markierungen und Register die Schlüssel-schalterstellung festgestellt werden und der Schlüsselschalter sich in der richtigen Stellung befinden (z.B. Aktivierung des zweiten Hauptschlüsselteils).

# CC:

- 1. erfolgreiche Operation
- 2. CKP1 oder CKP2 oder CKEK ist ungültig
- 3. fehlgeschlagene Operation (Fehler).

# Steuervektorprüfung: keine

#### Ersten Schlüsselteil laden (LFKP)

Beschreibung: Diese Anweisung chiffriert den ersten Teil eines Schlüssels, der im Schlüsselteileregister gespeichert ist. Der chiffrierte Schlüsselteil wird dann vom CFAP im Schlüsselspeicher gespeichert, von wo er später abgerufen und (durch die Anweisung "Schlüsselteile kombinieren") mit anderen Teilen des Schlüssels kombiniert werden kann. Bei der Ausführung dieser Anweisung wird die Markierung des Schlüsselteileregisters von "voll" auf "leer" gesetzt.

Damit diese Operation ausgeführt wird, muß das Schlüsselteileregister "voll" sein.

#### Schlüsselteile kombinieren (CKP)

- ☛ Beschreibung: Die Anweisung "Schlüsselteile kombinieren" kombiniert einen im Schlüsselteileregister gespeicherten Schlüsselteil mit dem vorausgehenden Schlüsselteil eines Schlüssels. Die Kombination der Schlüsselteile erfolgt durch XOR-Verknüpfung. Bei der Ausführung dieser Anweisung wird die Markierung des Schlüsselteileregisters von "voll" auf "leer" gesetzt.

Diese Operation wird nur ausgeführt, wenn das Schlüsselteileregister "voll" ist.

#### Prüfmuster berechnen (CVP)

Beschreibung: Die Anweisung berechnet ein Prüfmuster für einen vorgegebenen Schlüssel K. Mit dem Prüfmuster kann geprüft werden, ob ein manuell installierter Schlüssel richtig eingegeben wurde. Einzelheiten dieser Anweisung sprengen den Rahmen der vorliegenden Erfindung.

ANSI Fehlererkennungscode generieren (AGEDC)

- Gleichung:        A ====        EDC
- Eingabe:         A            Daten, für die der Fehlererkennungscode generiert werden soll.
- Ausgabe:        EDC          Für die durch A angegebenen Daten berechneter 64-Bit-Fehlererkennungscode.

Beschreibung: Fehlerbeschreibungscodes werden in ANSI X9.17-1985 zur Erkennung von Übertragungsfehlern oder Verarbeitungsfehlern verwendet, wenn andere Maßnahmen (z.B. Identifikationsüberprüfung unter einem geheimen Schlüssel) nicht möglich sind. Der Fehlererkennungscode wird mit Hilfe des in ANSI X9.9-1982 definierten Identifikationsüberprüfungsverfahrens (MAC) und eines feststehenden, nicht geheimen Schlüssels KDX = X'01234567890ABCDEF' generiert.

- ☛ AGECD kann im CFAP mit Hilfe der Anweisung "Codieren" und dem unchiffrierten Wert KDX simuliert werden. Diese Anweisung darf NICHT mittels anderer CA-Anweisungen (wie z.B. GMAC oder "Chiffrieren") mit einer unter dem Hauptschlüssel chiffrierten vorausgerechneten Form von KDX simuliert werden (da dies ein bekannter unter dem Hauptschlüssel chiffrierter Wert ist).

ANSI Fehlererkennungscode überprüfen (AVEDC)

- Gleichung: A, EDC ====    ja/nein

- Eingabe:           A           Daten, die mit dem übergebenen Fehlererkennungscode überprüft werden sollen.
- EDC       64-Bit-Fehlererkennungscode, der für die durch A angegebenen Daten überprüft werden soll.
- Ausgabe:       ja/nein   gibt an, ob der mit den durch A angegebenen Daten berechnete Fehlererkennungscode mit dem übergebenen EDC übereinstimmt.

Beschreibung: Fehlerbeschreibungscodes werden in ANSI X9.17-1985 zur Erkennung von Übertragungsfehlern oder Verarbeitungsfehlern verwendet, wenn andere Maßnahmen (z.B. Identifikationsüberprüfung unter einem geheimen Schlüssel) nicht möglich sind. Der Fehlererkennungscode wird mit Hilfe des in ANSI X9.9-1982 definierten Identifikationsüberprüfungsverfahrens (MAC) und eines feststehenden, nicht geheimen Schlüssels KDX = X'01234567-890ABCDEF' generiert.

AVECD kann im CFAP mit Hilfe der Anweisung "Codieren" und dem unchiffrierten Wert KDX simuliert werden. Diese Anweisung darf NICHT mittels anderer CA-Anweisungen (wie z.B. GMAC oder VMAC) mit einer unter dem Hauptschlüssel chiffrierten vorausberechneten Form von KDX simuliert werden (da dies ein bekannter unter dem Hauptschlüssel chiffrierter Wert ist).

#### ANSI Partiellen Beglaubigungsschlüssel erstellen (APNOTR)

- Gleichung:       Modus,e\*KM.C1L(KK1),e\*KM.C1R(KKr), FMID,  
                  TOID, C1L,C1R, C2L,C2R ====  
                  e\*KM.C2L(KKNIL),e\*KM.C2R(KKNIR)
- Eingabe:   Modus       gibt an, ob die Eingabe \*KK = KK1 //  
                          KKr ein verdoppelter 64-Bit-KEK (KK1 =  
                          KKr) oder ein echter 128-Bit-KEK ist.
- 0:   echter 128-Bit-KEK

-- 1: verdoppelter 64-Bit-KEK

Der Algorithmus zur Erstellung von partiellen Beglaubigungsschlüsseln ist bei 64-Bit-KEKs geringfügig anders als bei 128-Bit-KEKs (siehe "Beglaubigungsalgorithmen"). Der richtige Algorithmus wird durch den Modus-Parameter ausgewählt, da die richtige Auswahl nicht vom Eingabe-KEK selbst abgeleitet werden kann (sowohl 64-Bit-KEKs als auch 128-Bit-KEKs werden als 128 Bit eingegeben).

e*KM.C1L(KK1)	64-Bit-KK1, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C1L. KK1 ist die linke Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels *KK.
e*KM.C1R(KKr)	64-Bit-KKr, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C1R. KKr ist die rechte Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels *KK.

HINWEIS: In der CA werden alle KEKs (einschließlich ANSI KEKs) in 128-Bit-Form, als linke und rechte 64-Bit-Schlüsselteile, gespeichert. (64-Bit-KEKs werden zu 128 Bit verdoppelt. In diesem Fall ist die linke und die rechte Schlüsselhälfte identisch.) Der linke Teil von \*KK wird unter dem Hauptschlüssel in Verbindung mit einem linken Steuervektor chiffriert und im CKDS gespeichert. Entsprechend wird der rechte Teil von \*KK unter dem Hauptschlüssel in Verbindung mit einem rechten Steuervektor chiffriert und im CKDS gespeichert. Die Schlüsselformbits im Steuervektor unterscheiden zwischen linker und rechter Hälfte und zwischen 64- und 128-Bit-KEKs.

FMID            16 ASCII-Zeichen gemäß der Definition in  
ANSI X3.4-1977.



Dies ist eine "Ausgangsknoten"-Kennung, die als Beglaubigungsparameter verwendet wird. Ist sie keine 16 Zeichen lang, muß sie, wie in Abschnitt 7.5 von ANSI X9.17-1985, "Notarisation of Keys", beschrieben, wiederholt werden, bis 16 Zeichen erreicht sind. (Siehe Referenzliteraturliste).

TOID            16 ASCII-Zeichen gemäß der Definition in  
ANSI X3.4-1977.

Dies ist eine "Zielknoten"-Kennung, die als Beglaubigungsparameter verwendet wird. Ist sie keine 16 Zeichen lang, muß sie, wie in Abschnitt 7.5 von ANSI X9.17-1985, "Notarisation of Keys", beschrieben, wiederholt werden, bis 16 Zeichen erreicht sind. (Siehe Referenzliteraturliste).

C1L,C1R	64-Bit-Steuervektoren für linken bzw. rechten Teil von *KK.
C2L,C2R	64-Bit-Steuervektoren für linken bzw. rechten Teil von *KKNI.
-    Ausgabe: e*KM.C2L(KKNIL)	64-Bit-KKNIL, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C2L.

Dies ist die linke Hälfte eines 128-Bit-\*KKNI, einer partiellen oder zwischengeschalteten beglaubigenden Form des Eingabe-Schlüsselchiffrierschlüssels \*KK. Ein 'partieller' Beglaubigungsschlüssel ist ein Beglaubigungsschlüssel ohne Offset. Bevor \*KKNI als Beglaubigungsschlüssel für den Import oder Export von beglaubigten Schlüsseln benutzt werden kann, muß ein Offset durchgeführt werden. Dies geschieht implizit durch ARTMK, ARFMK und AXLTKEY.

e*KM.C2R(KKNIR)	64-Bit-KKNIR, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C2R.
-----------------	---

Dies ist die rechte Hälfte eines 128-Bit-\*KKNI, einer partiellen oder zwischengeschalteten beglaubigenden Form des Eingabe-Schlüsselchiffrierschlüssels \*KK.

HINWEIS: Alle KEKs und alle partiellen Beglaubigungs-KEKs werden in der CA in 128-Bit-Form gespeichert.

Beschreibung: Die ANSI-Beglaubigung ist ein Verfahren zur Versiegelung von Schlüsseln mit den Identitäten der beiden Kommunikationspartner, d.h. dem Sender und dem Empfänger der Schlüssel. Beglaubigte Schlüssel können nur rekonstruiert werden, wenn der ursprüngliche Schlüsselchiffrierschlüssel und die Identitäten der Kommunikationspartner bekannt sind. Ein Datenschlüssel oder ein Schlüsselchiffrierschlüssel kann vor der Übertragung beglaubigt werden, indem er mit Hilfe eines Beglaubigungsschlüssels (\*)KN chiffriert wird.

In ANSI wird (\*)KN durch EXKLUSIV-ODER-Verknüpfung eines KEK (\*)KK mit einem Beglaubigungssiegel NS gebildet, das aus der Identität des Schlüsselsenders, der Identität des gewünschten Schlüsselempfängers und dem aktuellen Wert eines dem (\*)KK zugeordneten Schlüsselnachrichtenzählers gebildet wird. Es ist zu beachten, daß (\*)KN für jede Übertragung neu berechnet werden muß, da der Zählerwert dynamisch ist, d.h. mit jeder Verwendung von (\*)KK anwächst.

In der CA wurde die Bildung von (\*)KN in zwei separate Schritte aufgeteilt. Zuerst wird APNOTR aufgerufen, um eine mit \*KKNI bezeichnete Zwischenform von (\*)KN zu berechnen, die gerade auf den statischen Werten von NS basiert, d.h. auf der Identität des Senders, der Identität des beabsichtigten Empfängers und (\*)KK selbst. Der zweite Schritt, der sog. Offset-Schritt, kombiniert \*KKNI und den \*KK zugeordneten aktuellen Zeigerwert zu (\*)KN. Für diese Aufspaltung in zwei getrennte Schritte gibt es zwei Gründe:

- 1. Effizienz. Da (\*)KN aus drei statischen Werten (den Identitäten der beiden Kommunikationspartner und (\*)KK selbst) und nur einem dynamischen Wert (dem Zählerwert, der (\*)KK zugeordnet ist) zusammengesetzt wird, besteht keine Notwendigkeit, (\*)KN bei jeder Aktualisierung des (\*)KK-Zählers komplett neu zu berechnen. Der Prozeß der Berechnung von (\*)KN kann deshalb in eine einmalig ausgeführte Berechnung mit Hilfe der statischen Werte und eine einfachere wiederholte Berechnung mit Hilfe des dynamischen Wertes aufgespaltet werden.
- 2. Transparenz. In ANSI müssen alle KEKs, ob mit oder ohne Beglaubigung, einer Offset-Operation unterzogen werden, bevor sie zur Chiffrierung eines Schlüssels (KD oder (\*)KK) für die Übertragung benutzt werden können. Die Offset-Operation wurde deshalb als implizite Operation in alle CA ANSI-Operationen integriert, bei denen ein KEK zur Chiffrierung oder Dechiffrierung eines anderen Schlüssels benutzt wird, d.h. bei den Anweisungen ARTMK, ARFMK und AXLTKEY. Durch die beschriebene Aufspaltung der (\*)KN-Erzeugung wird die Beglaubigung für die Anweisungen ARTMK, ARFMK und AXLTKEY transparent.

Mit APNOTR wird ein partieller Beglaubigungsschlüssel \*KKN1 für einen bestimmten KEK generiert. Partielle Beglaubigungsschlüssel werden verwendet, wenn ein beglaubigter Schlüssel (KD oder (\*)KK) durch ARTMK importiert, durch AXLTKEY umgewandelt oder durch ARFMK exportiert werden soll. Angenommen, ein Datenschlüssel DK1 soll von der Chiffrierung unter einem Offset-KEK \*KK1 unter die Beglaubigungsform eines anderen KEK, \*KK2, umchiffriert werden. In diesem Fall wird APNOTR mit \*KK2 aufgerufen und identifiziert den Aufrufenden und den beabsichtigten Empfänger, um \*KKN11, die partielle beglaubigende Form von \*KK2 zu erstellen. \*KK1 und \*KKN12 werden dann als Eingabe- bzw. Ausgabe-KEKs an AXLTKEY übergeben. AXLTKEY führt die Offset-Operation für \*KK1 und \*KKN12 mit den entsprechenden Zählerwerten durch und

benutzt die so entstandenen KEKs, um KD1 intern zu rekonstruieren und für die Ausgabe neu zu chiffrieren. (Es ist zu beachten, daß APNOTR nur einmal für \*KK2 aufgerufen werden muß; \*KKNI2 könnte im Schlüsselspeicher gespeichert sein, damit er später benutzt werden kann, wenn eine Beglaubigung erforderlich ist. Bei diesem Schlüsselverwaltungskonzept bleibt es jedoch dem Betreiber überlassen, ob APNOTR dynamisch oder für jeden (\*)KK einmal ausgeführt werden soll.) Wird in diesem Beispiel keine Beglaubigung unter \*KK2 benötigt, könnten \*KK1 und \*KK2 direkt an AXLTKEY übergeben worden sein, um die Offset-Berechnung für KD1 durchzuführen und KD1 zu rekonstruieren und neu zu chiffrieren.

Die Offset-Operation ist bei der Verwendung aller ANSI KEKs immer implizit enthalten. Deshalb ist die Offset-Funktion in die Hardware eingebettet und wird von den Anweisungen ARTMK, ARFMK und AXLTKEY in der Chiffriervorrichtung implizit ausgeführt. Dadurch erübrigt sich eine separate Anweisung für die Offset-Funktion, und die Transparenz von Beglaubigungsschlüsseln wird für diese Funktionen erhöht. APNOTR soll unter der Annahme, daß es einen separaten (\*)KK gibt, der von zwei kommunizierenden Knoten geteilt wird, und daß die Knotenkennungen sich nicht ändern, die Beglaubigungsleistung verbessern.

Der Algorithmus zur Bildung von partiellen Beglaubigungsschlüsseln ist in Fig. 48 und Fig. 49 dargestellt. Der Algorithmus soll die Schritte ausnutzen, die bei den Algorithmen für KEKs einfacher Länge bzw. für KEKs doppelter Länge gleich sind. Wenn der Eingabe-KEK und der ausgegebene partielle Beglaubigungs-KEK als 128-Bit-Schlüssel behandelt werden, besteht der einzige Unterschied in der Berechnung der Werte NSl und NSr, die durch EXKLUSIV ODER-Verknüpfung mit dem Eingabe-KEK den \*KKNI ergeben. Der Modus-Parameter, der die tatsächliche Schlüssellänge des Eingabe-KEK angibt, steuert die Berechnung von NSl und NSr. Ein Multiplexer wählt vom Modus-Parameter gesteuert eine von zwei 32-Bit-Eingaben aus, die zur Erzeugung der rechten

Hälfte von NSI benutzt werden. Entsprechend wählt ein Multiplexer eine von zwei Eingaben aus, die zur Erzeugung der linken Hälfte von NSr benutzt wird. Dieser Auswahlprozeß ist in Fig. 49 dargestellt. Die Verwendung des Modus-Parameters und der Multiplexer ermöglicht die Bildung eines \*KKNI, der nicht von der tatsächlichen Schlüssellänge des eingegebenen \*KK abhängig ist, und der mit dem in ANSI X9.17 definierten Beglaubigungsschlüssel (nach dem Offset) äquivalent und mit anderen CA ANSI KEKs speicherkompatibel ist.

Die Eingabe an APNOTR ist immer ein 128-Bit-KEK; die Ausgabe ist immer 128 Bit lang. Wenn an APNOTR ein verdoppelter 64-Bit-KEK übergeben wird, muß der Modus-Parameter auf 1 gesetzt sein. APNOTR gibt dann einen verdoppelten 64 Bit langen partiellen Beglaubigungs-KEK aus. Die Werte in den Schlüsselformfeldern von C1L, C1R, C2L und C2R müssen mit den Werten des übergebenen Modus-Parameters konsistent sein (siehe "Steuervektorprüfung"). Partielle Beglaubigungsschlüssel können nicht erneut als Eingabeschlüssel an APNOTR übergeben werden. Dieser Aspekt des Schlüsselverwaltungskonzepts wird dadurch erzwungen, daß die Hardware die Steuervektoren C1L, C1R, C2L und C2R sowie die linken und die rechten 64 Bit von \*KKNI prüft, und wird einmal aufgerufen, um einen partiellen Beglaubigungsschlüssel zu erstellen. Dies steht im Gegensatz zu anderen CA-Funktionen, bei denen für jede 64-Bit-Hälfte eines 128-Bit-KEK ein separater Funktionsaufruf erforderlich ist.

# CC:

- 1. erfolgreiche Operation
- 2. C1L oder C1R ist ungültig
- 3. C2L oder C2R ist ungültig
- 4. fehlgeschlagene Operation (Fehler).

# Steuervektorprüfung:

- 1. Prüfung auf C1L:
  - CV-Art = "KEK/ANSI"
  - APNOTR-Verwendungsbit = '1'
  - Schlüsselform (C1L) = Schlüsselform (C2L)
  - reserviert (48:63) = X'0'
- 2. Prüfung auf C1R:
  - CV-Art = "KEK/ANSI"
  - APNOTR-Verwendungsbit = '1'
  - Schlüsselform (C1R) = Schlüsselform (C2R)
  - reserviert (48:63) = X'0'
- 3. Prüfung auf C2L:
  - CV-Art = "KEK/ANSI"
  - APNOTR-Verwendungsbit = '0'
  - reserviert (48:63) = X'0'
- 4. Prüfung auf C2R:
  - CV-Art = "KEK/ANSI"
  - APNOTR-Verwendungsbit = '0'
  - reserviert (48:63) = X'0'

ANSI Umchiffrieren vom Hauptschlüssel (ARFMK)

- Gleichung:  $e*KM.C1L(KKL), e*KM.C1R(KKR), e*KM.C2(K), cntr,$   
Schlüsselart, C1L, C1R, C2, (C3)  
====  $e*KKo(K), e*KM.C3(K)$  (Schlüsselart = 0, Datenschlüssel)  
oder  
 $e*KKo(K)$  (Schlüsselart = 1, KEK)
- Eingabe:  $e*KM.C1L(KKL)$  linke 64 Bits von \*KK, als KKL bezeichnet, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C1L. \*KK ist ein 128-Bit-Schlüsselchiffrierschlüssel oder ein partieller Beglaubigungsschlüssel.  
  
 $e*KM.C1R(KKR)$  rechte 64 Bits von \*KK, als KKR bezeichnet, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C1R. \*KK ist ein 128-Bit-Schlüsselchiffrierschlüssel oder ein partieller Beglaubigungsschlüssel.

HINWEIS: In der CA werden alle KEKs (einschließlich ANSI KEKs) in 128-Bit-Form, als linke und rechte 64-Bit-Schlüsselteile, gespeichert. (64-Bit-KEKs werden zu 128 Bit verdoppelt. In diesem Fall ist die linke und die rechte Schlüsselhälfte identisch.) Der linke Teil von \*KK wird unter dem Hauptschlüssel in Verbindung mit einem linken Steuervektor chiffriert und im CKDS gespeichert. Entsprechend wird der rechte Teil von \*KK unter dem Hauptschlüssel in Verbindung mit einem rechten Steuervektor chiffriert und im CKDS gespeichert. Die Schlüsselformbits im Steuervektor unterscheiden zwischen linker und rechter Hälfte und zwischen 64- und 128-Bit-KEKs.

e\*KM.C2(K) zu exportierender 64-Bit-ANSI-Schlüssel K; K ist dreifach chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C2.

Dieser Schlüssel kann ein Datenschlüssel, ein 64-Bit-Schlüsselchiffrierschlüssel oder die linke oder rechte Hälfte eines 128-Bit-Schlüsselchiffrierschlüssels sein.

cntr	64 Bit langer unchiffrierter Sendezählerwert, der dem Schlüsselchiffrierschlüssel *KK zugeordnet ist.
------	---

Dieser Wert wird gewöhnlich zusammen mit dem exportierten Schlüssel an den vorgesehenen Empfänger übertragen. Der Empfänger verwendet den empfangenen Zählerwert zur Erkennung von Wiederholungen, zum Synchronisieren seines lokalen Empfangszählers für \*KK und zur Rekonstruktion des exportierten Schlüssels. Lokale Zähler werden vom CFAP mit Integrität verwaltet.

Schlüsselart	Art des zu exportierenden Schlüssels
--------------	--------------------------------------

HINWEIS: Für Datenschlüssel existieren zwei Formen der chiffrierten Ausgabe von ARFMK. Eine Form der Ausgabe ist für den Export an den vorgesehenen Empfänger bestimmt; der Schlüssel wird unter einer Offset-Form des angegebenen Schlüsselchiffrierschlüssels chiffriert. Die andere Form kann entweder direkt für MAC CSMs verwendet werden (wenn ein einzelner KD exportiert werden soll) oder später mit einem anderen solchen Schlüssel zu einem CSM MAC-Schlüssel kombiniert werden (wenn zwei KDs gleichzeitig exportiert werden sollen). Zum Kombinieren dieser 'partiellen' MAC-Schlüssel in einen CSM MAC-Schlüssel wird die Anweisung ACOMBKD verwendet.



-- 0: KD  
-- 1: KK

C1L,C1R 64-Bit-Steuervektoren für die linke bzw. rechte Hälfte des Schlüsselchiffrierschlüssels oder des partiellen Beglaubigungsschlüssels \*KK.

C2 64-Bit-Steuervektor für den Eingangsschlüssel K.

C3 64-Bit-Steuervektor für den CSM MAC-Schlüssel oder partiellen CSM MAC-Schlüssel, der unter KM gespeichert werden soll. Diese Eingabe ist nur für Schlüsselart = 0 gültig.

- Ausgaben: e\*KKo(K) 64-Bit-Schlüssel K, dreifach chiffriert unter dem Schlüsselchiffrierschlüssel oder Beglaubigungsschlüssel \*KKo, wobei \*KKo der Schlüssel \*KK nach Offset mit cntr ist.

Ist \*KK ein partieller Beglaubigungsschlüssel, angegeben durch die Notation \*KKNI, so ist \*KKo ein Beglaubigungsschlüssel mit der Notation \*KN, und K ist zu 'beglaubigen'.

e\*KM.C3(K) 64-Bit-Datenschlüssel K, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C3.

Diese Ausgabe ist nur für Schlüsselart = 0 gültig. Sie wird entweder direkt für MAC CSMS oder mittels ACOMBKD mit einem anderen solchen Schlüssel für MAC CSMS kombiniert.

cntr 64 Bit langer unchiffrierter Sende-  
zählerwert, der dem Schlüs-

selchiffrierschlüssel \*KK zugeordnet  
ist.

Dieser Wert wird gewöhnlich zusammen mit dem exportierten Schlüssel an den vorgesehenen Empfänger übertragen. Der Empfänger verwendet den empfangenen Zählerwert zur Erkennung von Wiederholungen, zum Synchronisieren seines lokalen Empfangszählers für \*KK und zur Rekonstruktion des exportierten Schlüssels. Lokale Zähler werden vom CFAP mit Integrität verwaltet.

Schlüsselart     Art des zu exportierenden Schlüssels

HINWEIS: Für Datenschlüssel existieren zwei Formen der chiffrierten Ausgabe von ARFMK. Eine Form der Ausgabe ist für den Export an den vorgesehenen Empfänger bestimmt; der Schlüssel wird unter einer Offset-Form des angegebenen Schlüsselchiffrierschlüssels chiffriert. Die andere Form kann entweder direkt für MAC CSMS verwendet werden (wenn ein einzelner KD exportiert werden soll) oder später mit einem anderen solchen Schlüssel zu einem CSM MAC-Schlüssel kombiniert werden (wenn zwei KDs gleichzeitig exportiert werden sollen). Zum Kombinieren dieser 'partiellen' MAC-Schlüssel in einen CSM MAC-Schlüssel wird die Anweisung ACOMBKD verwendet.

-- 0: KD  
-- 1: KK

C1L,C1R	64-Bit-Steuervektoren für die linke bzw. rechte Hälfte des Schlüsselchiffrierschlüssels oder des partiellen Beglaubigungsschlüssels *KK.
C2	64-Bit-Steuervektor für den Eingabeschlüssel K.

C3                    64-Bit-Steuervektor für den CSM MAC-Schlüssel oder partiellen CSM MAC-Schlüssel, der unter KM gespeichert werden soll. Diese Eingabe ist nur für Schlüsselart = 0 gültig.

- Ausgabe:        e\*KKo(K)        64-Bit-Schlüssel K, dreifach chiffriert unter dem Schlüsselchiffrierschlüssel oder Beglaubigungsschlüssel \*KKo, wobei \*KKo der Schlüssel \*KK nach Offset mit cntr ist.

Ist \*KK ein partieller Beglaubigungsschlüssel, angegeben durch die Notation \*KKNI, so ist \*KKo ein Beglaubigungsschlüssel mit der Notation \*KN, und K ist zu 'beglaubigen'.

e\*KM.C3(K)        64-Bit-Datenschlüssel K, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C3.

Diese Ausgabe ist nur für Schlüsselart = 0 gültig. Sie wird entweder direkt für MAC CSMS oder mittels ACOMBKD mit einem anderen solchen Schlüssel für MAC CSMS kombiniert.

Beschreibung: Die Anweisung ARFMK chiffriert einen unter dem Hauptschlüssel chiffrierten 64-Bit-Schlüssel K unter einen 128-Bit-Schlüsselchiffrierschlüssel oder einen verdoppelten 64-Bit-Schlüsselchiffrierschlüssel \*KK (den sog. Exportschlüssel) um. Ein 128-Bit-Schlüssel K kann exportiert werden, indem für jede Hälfte die Anweisung ARFMK aufgerufen wird; \*KK muß in diesem Fall ein echter 128-Bit-KEK sein. Diese Regel wird durch die Hardware erzwungen (siehe "Steuervektorprüfung").

Mit ARFMK können Schlüssel entweder in beglaubigter oder in unbeglaubigter Form gemäß ANSI X9.17 exportiert werden. ANSI-

Schlüssel können in nichtbeglaubigter Form exportiert werden, indem ARFMK mit einem Schlüsselchiffrierschlüssel aufgerufen wird, den der Sender und der Empfänger besitzen. ARFMK führt intern eine Offset-Operation für den angegebenen Schlüsselchiffrierschlüssel durch. In beglaubigter Form können ANSI-Schlüssel exportiert werden, indem ARFMK mit der partiellen beglaubigenden Form eines Schlüsselchiffrierschlüssels aufgerufen wird, den sowohl der Sender als auch der Empfänger besitzen. ARFMK führt intern eine Offset-Operation für den angegebenen partiellen Beglaubigungsschlüssel durch, um den Beglaubigungsschlüssel zu erstellen. Partielle Beglaubigungsschlüssel werden mit der Anweisung APNOTR aus Schlüsselchiffrierschlüsseln gebildet.

Angenommen, \*KKab ist ein Schlüsselchiffrierschlüssel, den die Knoten A und B besitzen, und \*KKNIab ist ein partieller Beglaubigungsschlüssel, der durch APNOTR aus \*KKab gebildet wurde. In diesem Fall kann der Schlüssel K in unbeglaubigter Form von Knoten A nach Knoten B exportiert werden, indem ARFMK mit \*KKab als Schlüsselchiffrierschlüssel aufgerufen wird. Der Schlüssel K kann in beglaubigter Form von A nach B exportiert werden, indem ARFMK mit \*KKNIab als Schlüsselchiffrierschlüssel aufgerufen wird.

- In ANSI X9.17 können ein oder zwei KDs in einer einzigen CSM exportiert werden. Wenn nur ein einziger KD versendet wird, kann er schließlich entweder als Vertraulichkeitsschlüssel oder als MAC-Schlüssel benutzt werden. Die CSM selbst wird mit dem gesendeten KD einer Identifikationsüberprüfung unterzogen. Werden zwei KDs versendet, ist der erste KD ein MAC-Schlüssel und der zweite ein Vertraulichkeitsschlüssel. Die CSM wird einer Identifikationsüberprüfung durch EXKLUSIV-ODER-Verknüpfung der beiden KDs unterzogen.

ARFMK unterstützt die CSM-Identifikationsüberprüfung durch Ausgabe exportierter Datenschlüssel in zwei Formen. Die erste Form

des KD ist für den Export bestimmt, d.h. sie wird unter einer Offset-Form des angegebenen Schlüsselchiffrierschlüssels chiffriert. Die zweite Form des KD kann lokal für eine direkte MAC-Operation der CSM (wie beim Export eines einzelnen KD) verwendet oder mit einem anderen KD dieser Form zur Erstellung eines Schlüssels für die MAC-Operation der CSM kombiniert werden (durch EXKLUSIV ODER). Die letztere Verwendung entspricht dem Fall, daß zwei KDs in einer einzigen CSM zusammengefaßt werden. Der Parameter C3 entscheidet, ob die zweite Ausgabeform ein MAC-Schlüssel oder ein 'partieller' MAC-Schlüssel ist. Zum Kombinieren dieser 'partiellen' MAC-Schlüssel in einen einzigen CSM MAC-Schlüssel wird die Anweisung ACOMBKD verwendet.

Fig. 50 ist das Blockdiagramm der Anweisung ARFMK.

# CC:

- 1. erfolgreiche Operation
- 2. C1L oder C1R ist ungültig
- 3. C2 ist ungültig
- 4. fehlgeschlagene Operation (Fehler).

# Steuervektorprüfung:

- 1. Prüfung von C1L:

-- CV-Art	=	"KEK/ANSI"
-- ARFMK-Verwendungsbit	=	'1'
-- reserviert (48:63)	=	X'0'

- 2. Prüfung von C1R:

-- CV-Art	=	"KEK/ANSI"
-- ARFMK-Verwendungsbit	=	'1'
-- reserviert (48:63)	=	X'0'

- 3. Prüfung von C2:

-- CV-Art = "KEK/ANSI" oder "Daten/ANSI"  
-- Exportkontrollbit 1 = 0 (RFMK zulässig)  
-- Falls CV-Art = "Daten/ANSI", ist folgende  
Prüfung durchzuführen:

-- In Fig. 51 sind die zulässigen Kombinationen von zu prüfenden C2-Attributen aufgeführt. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig. E, D, MG, MV, ACMB sind die Verwendungsbits für den Datenschlüssel-Steuervektor.

- 4. Prüfung von C2 & C1L,C1R:

-- In Fig. 52 sind die zulässigen Kombinationen von C2-Art und -Schlüsselform und Schlüsselformattributen für C1L und C1R aufgeführt. Diese Attribute werden geprüft, um die Trennung der beiden Schlüsselhälften zu erzwingen und um sicherzustellen, daß ein echter 128-Bit-KEK nur unter einem 128-Bit-KEK exportiert werden kann. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig.

- 5. Prüfung von C3:

-- CV-Art = "Daten/ANSI"  
-- reserviert (48:63) = X'0'  
-- Exportkontrollbit 1 = 1 (kein Export)

-- In Fig. 53 sind die zulässigen Kombinationen von zu prüfenden C3-Attributen aufgeführt. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig. E, D, MG, MV, ACMB sind die Verwendungsbits für den Datenschlüssel-Steuervektor.

ANSI Umchiffrieren unter den Hauptschlüssel (ARTMK)

- Gleichung:  $e*KM.C1L(KKL), e*KM.C1R(KKR), e*KKo(K), cntr,$   
Schlüsselart, C1L, C1R, C2, (C3)  
====  $e*KM.C2(K), e*KM.C3(K)$  (Schlüsselart = 0,  
Datenschlüssel)  
oder  
 $e*KM.C2(K)$  (Schlüsselart = 1, KEK)
- Eingabe:  $e*KM.C1L(KKL)$  linke 64 Bits von \*KK, als KKL  
bezeichnet, chiffriert unter dem  
Hauptschlüssel in Verbindung mit  
einem Steuervektor C1L.

\*KK ist ein 128-Bit-Schlüsselchiffrierschlüssel oder ein partieller Beglaubigungsschlüssel.

$e*KM.C1R(KKR)$  rechte 64 Bits von \*KK, als KKR  
bezeichnet, chiffriert unter dem  
Hauptschlüssel in Verbindung mit  
einem Steuervektor C1R.

\*KK ist ein 128-Bit-Schlüsselchiffrierschlüssel oder ein partieller Beglaubigungsschlüssel.

HINWEIS: In der CA werden alle KEKs (einschließlich ANSI KEKs) in 128-Bit-Form, als linke und rechte 64-Bit-Schlüsselteile, gespeichert. (64-Bit-KEKs werden zu 128 Bit verdoppelt. In diesem Fall ist die linke und die rechte Schlüsselhälfte identisch.) Der linke Teil von \*KK wird unter dem Hauptschlüssel in Verbindung mit einem linken Steuervektor chiffriert und im CKDS gespeichert. Entsprechend wird der rechte Teil von \*KK unter dem Hauptschlüssel in Verbindung mit einem rechten Steuervektor chiffriert und im CKDS gespeichert. Die Schlüsselformbits im Steuervektor unterscheiden zwischen linker und rechter Hälfte und zwischen 64- und 128-Bit-KEKs.

**e\*KKo(K)** zu importierender 64-Bit-ANSI-Schlüssel K; K ist dreifach chiffriert unter dem Schlüsselchiffrierschlüssel oder dem Beglaubigungsschlüssel \*KKo, wobei \*KKo der Schlüssel \*KK nach Offset mit cntr ist.

Ist \*KK ein partieller Beglaubigungsschlüssel, angegeben durch die Notation \*KKNI, so ist \*KKo ein Beglaubigungsschlüssel mit der Notation \*KN, und K ist zu beglaubigen. Es ist zu beachten, daß wenn \*KK ein verdoppelter 64-Bit-Schlüssel ist, d.h. wenn \*KK = KK//KK ist, e\*KKo(K) äquivalent mit eKKo(K), einem einfach unter einem Offset-64-Bit-Schlüssel chiffrierten ANSI-Schlüssel K, ist.

**cntr** 64 Bit langer unchiffrierter Zählerwert, der dem Schlüsselchiffrierschlüssel \*KK zugeordnet ist.

Dieser Wert wird gewöhnlich vom Sender übermittelt und ist der Zählerwert für den Offset von \*KK vor der Chiffrierung von K. Cntr muß gemäß dem Zählerverwaltungskonzept nach ANSI X9.17 mit dem entsprechenden lokalen Empfangszähler für \*KK verglichen werden, bevor der empfangene Schlüssel mit ARTMK importiert wird. Der Vergleichsschritt und der ARTMK-Aufrufschritt müssen im CFAP autark ausgeführt werden, damit die Integrität gewahrt bleibt. Die lokalen Zähler werden vom CFAP mit Integrität verwaltet.

**Schlüsselart** Art des zu importierenden Schlüssels

**HINWEIS:** Für Datenschlüssel existieren zwei Formen der chiffrierten Ausgabe von ARFMK. Eine Form der Ausgabe ist für die



letztendliche Verwendung des importierten Schlüssels bestimmt. Die andere Form kann entweder direkt für MAC CSMs verwendet werden (wenn ein einzelner KD importiert werden soll) oder später mit einem anderen solchen Schlüssel zu einem CSM MAC-Schlüssel kombiniert werden (wenn zwei KDs gleichzeitig importiert werden sollen). Zum Kombinieren dieser 'partiellen' MAC-Schlüssel in einen CSM MAC-Schlüssel wird die Anweisung ACOMBKD verwendet.

-- 0: KD  
-- 1: KK

C1L,C1R      64-Bit-Steuervektoren für die linke bzw. rechte Hälfte des Schlüsselchiffrierschlüssels oder des partiellen Beglaubigungsschlüssels \*KK.

C2            64-Bit-Steuervektor für den Schlüssel K, der unter KM gespeichert werden soll.

Dieser Steuervektor gibt den letztendlichen Verwendungszweck des importierten Schlüssels K an.

C3            64-Bit-Steuervektor für den CSM MAC-Schlüssel oder partiellen CSM MAC-Schlüssel, der unter KM gespeichert werden soll.

Diese Eingabe ist nur für Schlüsselart = 0 gültig.

- Ausgabe:      e\*KM.C2(K)    empfangener 64-Bit-Schlüssel K, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C2.
- e\*KM.C3(K)    64-Bit-Datenschlüssel K, dreifach chiffriert unter dem Hauptschlüssel

KM in Verbindung mit einem Steuervektor C3.

Diese Ausgabe ist nur für Schlüsselart = 0 gültig. Sie wird entweder direkt für MAC CSMs oder mittels ACOMBKD mit einem anderen solchen Schlüssel für MAC CSMs kombiniert.

Beschreibung: Die Anweisung ARTMK chiffriert einen unter einem 128-Bit-Schlüsselchiffrierschlüssel oder einem verdoppelten 64-Bit-Schlüsselchiffrierschlüssel \*KK (dem sog. Importschlüssel) chiffrierten 64-Bit-Schlüssel K unter den Hauptschlüssel um. Ein 128-Bit-Schlüssel K kann importiert werden, indem für jede Hälfte die Anweisung ARTMK aufgerufen wird.

Mit ARTMK können Schlüssel entweder in beglaubigter oder in unbeglaubigter Form gemäß ANSI X9.17 importiert werden. ANSI-Schlüssel können in nichtbeglaubigter Form importiert werden, indem ARTMK mit einem Schlüsselchiffrierschlüssel aufgerufen wird, den der Sender und der Empfänger besitzen. ARTMK führt intern mit dem angegebenen Zählerwert eine Offset-Operation für den angegebenen Schlüsselchiffrierschlüssel durch. In beglaubigter Form können ANSI-Schlüssel importiert werden, indem ARTMK mit der partiellen beglaubigenden Form eines Schlüsselchiffrierschlüssels aufgerufen wird, den auch der Sender des Schlüssels besitzt. ARTMK führt intern mit dem angegebenen Zählerwert eine Offset-Operation für den angegebenen partiellen Beglaubigungsschlüssel durch, um den Beglaubigungsschlüssel zu erstellen. Partielle Beglaubigungsschlüssel werden mit der Anweisung APNOTR aus Schlüsselchiffrierschlüsseln gebildet.

In ANSI X9.17 können ein oder zwei KDs in einer einzigen CSM exportiert werden. Wenn nur ein einziger KD empfangen wird, kann er schließlich entweder als Vertraulichkeitsschlüssel oder als MAC-Schlüssel benutzt werden. Die CSM selbst wird mit dem gesendeten KD einer Identifikationsüberprüfung unterzogen. Werden

zwei KDs empfangen, ist der erste KD ein MAC-Schlüssel und der zweite ein Vertraulichkeitsschlüssel. Die CSM wird einer Identifikationsüberprüfung durch EXKLUSIV-ODER-Verknüpfung der beiden KDs unterzogen.

ARTMK unterstützt die CSM-Identifikationsüberprüfung durch Ausgabe importierter Datenschlüssel in zwei Formen. Die erste Form des KD ist für den letztlichen Verwendungszweck bestimmt, d.h. je nach C2-Parameter entweder als Vertraulichkeitsschlüssel oder als MAC-Schlüssel. Die zweite Form des KD kann für eine direkte MAC-Operation der CSM (wie bei einem einzelnen KD) verwendet oder mit einem anderen KD dieser Form zur Erstellung eines Schlüssels für die MAC-Operation der CSM kombiniert werden (durch EXKLUSIV ODER). Die letztere Verwendung entspricht dem Fall, daß zwei KDs in einer einzigen CSM zusammengefaßt werden. Der Parameter C3 entscheidet, ob die zweite Ausgabeform ein MAC-Schlüssel oder ein 'partieller' MAC-Schlüssel ist. Zum Kombinieren dieser 'partiellen' MAC-Schlüssel in einen einzigen CSM MAC-Schlüssel wird die Anweisung ACOMBKD verwendet.

Fig. 54 ist das Blockdiagramm der Anweisung ARTMK.

# CC:

- 1. erfolgreiche Operation
- 2. C1L oder C1R ist ungültig
- 3. C2 ist ungültig
- 4. C3 ist ungültig
- 5. fehlgeschlagene Operation (Fehler).

# Steuervektorprüfung:

- 1. Prüfung von C1L:

-- CV-Art = "KEK/ANSI"

MA9-88-011

- ARTMK-Verwendungsbit = '1'
- reserviert (48:63) = X'0'
- 2. Prüfung von C1R:
  - CV-Art = "KEK/ANSI"
  - ARTMK-Verwendungsbit = '1'
  - reserviert (48:63) = X'0'
- 3. Prüfung von C2:
  - CV-Art = "KEK/ANSI" oder "Daten/ANSI"
  - ARTMK-Verwendungsbit = '1'
  - Falls CV-Art = "Daten/ANSI", ist folgende Prüfung durchzuführen:
    - In Fig. 55 sind die zulässigen Kombinationen von zu prüfenden C2-Attributen aufgeführt. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig. E, D, MG, MV, ACMB sind die Verwendungsbits für den Datenschlüssel-Steuervektor.
- 4. Prüfung von C2 & C1L,C1R:
  - In Fig. 56 sind die zulässigen Kombinationen von C2-Art und -Schlüsselform und Schlüsselformattributen für C1L und C1R aufgeführt. Diese Attribute werden geprüft, um die Trennung der beiden Schlüsselhälften zu erzwingen und um sicherzustellen, daß ein echter 128-Bit-KEK nur unter einem 128-Bit-KEK importiert werden kann. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig.
- 5. Prüfung von C3:

- CV-Art = "Daten/ANSI"
- reserviert (48:63) = X'0'
- Exportkontrollbit 1 = 1 (kein Export)
  
- In Fig. 57 sind die zulässigen Kombinationen von zu prüfenden C3-Attributen aufgeführt. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig. E, D, MG, MV, ACMB sind die Verwendungsbits für den Datenschlüssel-Steuervektor.

ANSI Schlüssel umwandeln (AXLTKEY)

- Gleichung:  $e*KM.C1L(KK1L), e*KM.C1R(KK1R), e*KM.C2L(KK2L), e*KM.C2R(KK2R), e*KK1o(*)Ko(KDmac), cntr1, cntr2, \text{Schlüsselart}, C1L, C1R, C2L, C2R, C3$   
====  $E*KK2o(K), e*KM.C3(K)$  (Schlüsselart = 0, KD)  
oder  
 $e*KK2o(*K), e*KM.C3(KDmac)$  (Schlüsselart = 1, KEK)
- Eingabe:  $e*KM.C1L(KK1L)$  linke 64 Bit von \*KK1, bezeichnet als KK1L, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C1L.

\*KK1 ist ein 128 Bit langer KEK oder partieller Beglaubigungs-KEK.

$e*KM.C1R(KK1R)$  rechte 64 Bit von \*KK1, bezeichnet als KK1R, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C1R.

\*KK1 ist ein 128 Bit langer KEK oder partieller Beglaubigungs-KEK.

HINWEIS: \*KK1 wird durch Offset mit dem Zählerwert cntrl intern zu \*KK1o, dem Eingabe-Schlüsselchiffrierschlüssel. Wird der umzuwandelnde Schlüssel in beglaubigter Form eingegeben, muß \*KK1 ein partieller Beglaubigungsschlüssel mit der Notation \*KKNI1 sein, der mit der Anweisung APNOTR erstellt wird. \*KK1o ist in diesem Fall ein Beglaubigungsschlüssel.

e*KM.C2L(KK2L)	linke 64 Bit von *KK2, bezeichnet als KK2L, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C2L.
----------------	---

\*KK2 ist ein 128 Bit langer KEK oder partieller Beglaubigungs-KEK.

e*KM.C2R(KK2R)	rechte 64 Bit von *KK2, bezeichnet als KK2R, chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C2R.
----------------	--

\*KK2 ist ein 128 Bit langer KEK oder partieller Beglaubigungs-KEK.

➤ HINWEIS: \*KK2 wird durch Offset mit dem Zählerwert cntr2 intern zu \*KK2o, dem Ausgabe-Schlüsselchiffrierschlüssel. Wird der umzuwandelnde Schlüssel in beglaubigter Form eingegeben, muß \*KK2 ein partieller Beglaubigungsschlüssel mit der Notation \*KKNI2 sein, der mit der Anweisung APNOTR erstellt wird. \*KK1o ist in diesem Fall ein Beglaubigungsschlüssel.

e*KK1o((*)K)	umzuwandelnder 64 oder 128 Bit langer ANSI-Schlüssel (KD oder KEK) mit der Bezeichnung (*)K.
--------------	--

(\*)K ist dreifach chiffriert unter \*KK1o, wobei \*KK1o einem Offset mit cntrl unterzogen wurde.

cntrl                      Wenn (\*)K 128 Bit lang ist, d.h. wenn  $i* = K1//Kr$  ist, wird dieser Parameter als  $e*KK1o(K1)//e*KK1o(Kr)$  eingegeben. Andernfalls, wenn (\*)K nur 64 Bit lang ist, wird dieser Parameter als  $e*KK1o(K)$  eingegeben.

$e(*)Ko(KDmac)$             optionaler 64 Bit langer MAC-Schlüssel KDmac, chiffriert unter  $(K)Ko$ ; d.h., einfach chiffriert unter einem 64-Bit-K mit Offset durch Null, oder dreifach chiffriert unter einem 128-Bit-\*K mit Offset durch Null.

In diesem Fall, wenn  $(*)Ko = (*)K$  ist, ist  $e(*)Ko(KDmac) = e(*)K(KDmac)$ . KDmac ist ein temporärer MAC-Schlüssel, der in ANSI X9.17 zur Identifikationsüberprüfung von Chiffrierservice-Meldungen (CSM) in einem Schlüsselumwandlungszentrum (KTC) benutzt wird. Ein KDmac begleitet immer eine CSM-Anforderung zur Umwandlung eines KEK im KTC. Eine CSM-Anforderung zur Umwandlung von KDs wird nicht von einem KDmac begleitet, da die Datenschlüssel selber als MAC-Schlüssel zur Identifikationsüberprüfung der CSMs vom bzw. an das KTC benutzt werden. Dieser Parameter ist deshalb nur für Schlüsselart = 1 gültig, d.h., wenn (\*)K ein KEK ist.

cntrl                      64 Bit langer unchiffrierter Zählerwert, der dem Schlüsselchiffrierschlüssel \*KK1 zugeordnet ist.

Dieser Wert wird gewöhnlich vom Sender übermittelt und ist der Zählerwert für den Offset von \*KK1 vor der Chiffrierung von (\*)K. Cntrl muß gemäß dem Zählerverwaltungskonzept nach ANSI X9.17 mit dem entsprechenden lokalen Empfangszähler für \*KK verglichen werden, bevor der empfangene Schlüssel mit AXLTKEY umgewandelt wird. Der Vergleichsschritt und der AXLTKEY-Aufrufschritt müssen im CFAP autark ausgeführt werden, damit die Integrität gewahrt bleibt. KEK-Zähler werden vom CFAP mit Integrität verwaltet.

cntr2	64 Bit langer unchiffrierter Senderzählerwert, der dem Schlüsselchiffrierschlüssel *KK2 zugeordnet ist.
-------	---

Dieser Wert wird gewöhnlich zusammen mit dem umgewandelten Schlüssel an den vorgesehenen Empfänger übermittelt. Der Empfänger verwendet den empfangenen Zählerwert zur Erkennung von Wiederholungen, zum Synchronisieren seines lokalen Empfangszählers für \*KK2 und zur Rekonstruktion des umgewandelten Schlüssels. Lokale Zähler werden vom CFAP mit Integrität verwaltet.

Schlüsselart	Art des umzuwandelnden Schlüssels
-- 0: KD	
-- 1: (*)KK, d.h. 64- oder 128-Bit-KEK	
C1L,C1R	64-Bit-Steuervektoren für die linke bzw. rechte Hälfte des Eingabe-Schlüsselchiffrierschlüssels oder partiellen Beglaubigungsschlüssels.
C2L,C2R	64-Bit-Steuervektoren für die linke bzw. rechte Hälfte des Ausgabe-Schlüsselchiffrierschlüssels oder partiellen Beglaubigungsschlüssels.



C3                    64-Bit-Steuervektor für den CSM MAC-Schlüssel oder partiellen MAC-Schlüssel, der unter KM gespeichert werden soll.

Soll ein einziger KEK oder genau ein einziger KD umgewandelt werden, muß C3 die MACGEN- und MACVER-Attribute enthalten. Sollten zwei KDs umgewandelt werden, muß C3 das ACOMBKD-Attribut enthalten, um einen 'partiellen' MAC-Schlüssel zu erstellen. Für die Umwandlung von zwei KDs sind zwei AXLTKEY-Aufrufe erforderlich. Die beiden so erzeugten partiellen MAC-Schlüssel können mit der Anweisung ACOMBKD zu einem CSM MAC-Schlüssel mit MACGEN- und MACVER-Attributen kombiniert werden.

Ausgabe:    e\*KK2o((\*)K)    64- oder 128-Bit-Schlüssel (K), dreifach chiffriert unter \*KK2o, ist der Ausgabe-Schlüsselchiffrierschlüssel \*KK2 mit Offset durch Zählerwert cntr2.

Wenn Schlüsselart = 0 ist, ist (K) ein 64-Bit-Datenschlüssel (d.h. (K) = K), und dieser Parameter wird mit e\*KK2o(K) bezeichnet. (K) wird von \*KK1o in \*KK2o umgewandelt.

e\*KM.C3(K)            64 Bit langer MAC-Schlüssel oder partieller MAC-Schlüssel, der mit dem umgewandelten Datenschlüssel K identisch ist, dreifach chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C3.

Diese Ausgabe ist nur gültig, wenn ein KD umgewandelt wird, d.h., wenn Schlüsselart = 0 ist. In ANSI X9.17 wird ein umzuwandelnder KD auch zur Identifikationsüberprüfung der Chiffrierservice-Meldungen an das bzw. vom KTC benutzt. Soll nur ein einziger KD umgewandelt werden, so ist der KD selber der MAC-Schlüssel.

sel. In diesem Fall ist  $e*KM.C3(K)$  genau der KD, K, in einer Form, die von GMAC und VMAC direkt für CSM MAC-Operationen verwendet werden kann. Sollen zwei KDs umgewandelt werden, wird der MAC-Schlüssel durch EXKLUSIV-ODER-Verknüpfung der beiden KDs gebildet. In diesem Fall muß AXLTKEY zweimal aufgerufen werden, nämlich für jeden KD einmal.  $e*KM.C3(K)$  von jedem Aufruf ist dann ein partieller MAC-Schlüssel. Die beiden partiellen MAC-Schlüssel können mit der Anweisung ACOMBKD zu einem Schlüssel kombiniert werden, der von GMAC und VMAC direkt für die CSM MAC-Operation benutzt werden kann.

$e*KM.C3(KDmac)$  64 Bit langer MAC-Schlüssel KDmac, dreifach chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor C3.

Diese Ausgabe ist nur bei der Umwandlung eines KEK gültig, d.h. wenn Schlüsselart = 1 ist. In ANSI X9.17 wird KDmac zur Identifikationsüberprüfung von CSMS vom bzw. an das KTC verwendet.

Beschreibung: Die Anweisung AXLTKEY chiffriert einen unter dem Offset-Schlüsselchiffrierschlüssel \*KK1 chiffrierten 64- oder 128-Bit-Schlüssel (\*)K unter einen Offset-Schlüsselchiffrierschlüssel \*KK2 um. Neben der Offset-Funktion unterstützt AXLTKEY die Umwandlung von bzw. in beglaubigte Schlüsselformen. Wenn (\*)K in beglaubigter Form eingegeben wird, muß \*KK1 ein partieller Beglaubigungsschlüssel sein. Entsprechend muß \*KK2 ein partieller Beglaubigungsschlüssel sein, wenn (\*)K in beglaubigter Form ausgegeben werden soll. Partielle Beglaubigungsschlüssel werden durch die Anweisung APNOTR aus KEKs gebildet.

AXLTKEY erzeugt auch eine sekundäre Ausgabe: einen MAC-Schlüssel oder einen MAC-Schlüsselteil, der zur Identifikationsüberprüfung von Chiffrierservice-Meldungen zwischen der Stelle, die die Umwandlung anfordert, und dem KTC, verwendet werden kann. Die Bil-

derung des MAC-Schlüssels wird durch den Schlüsselart-Parameter gesteuert.

Bei Schlüsselart = 1, d.h. KEK-Umwandlung, wird der MAC-Schlüssel der Anweisung als optionaler Parameter KDmac, chiffriert unter dem KEK selbst (Offset durch 0), übergeben. AXLTKEY rekonstruiert KDmac intern und gibt ihn unter KM in Verbindung mit einem durch den Parameter C3 angegebenen Steuervektor neu chiffriert aus. C3 wird typischerweise mit GMAC- und VMAC-Verwendungsattributen angegeben; die Hardware erzwingt, daß kein Export zugelassen wird.

Bei Schlüsselart = 0, d.h. KD-Umwandlung, basiert der MAC-Schlüssel auf den umzuwandelnden KDs. AXLTKEY rekonstruiert den KD intern und gibt ihn unter KM mit durch C3 festgelegten Attributen chiffriert aus. Soll genau ein KD umgewandelt werden, ist KD selbst der MAC-Schlüssel, und C3 wird typischerweise mit GMAC- und VMAC-Verwendungsattributen angegeben. Auch hier erzwingt die Hardware eine Exportkontrolle. Wenn zwei KDs, KD1 und KD2 umgewandelt werden sollen, muß AXLTKEY zweimal aufgerufen werden; der MAC-Schlüssel ist  $KD1 \text{ XOR } KD2$ , und C3 muß mit ACOMBKD-Attributen angegeben werden. Die erzeugten MAC-Schlüsselteile  $e * KM.C3(KD1)$  und  $e * (KM.C3(KD2))$  können dann an die Anweisung ACOMBKD übergeben werden, die sie intern rekonstruiert und den erforderlichen MAC-Schlüssel  $e * KM.Cx(KD1 \text{ XOR } KD2)$  erzeugt.

Fig. 58 und 59 sind Blockdiagramme der Anweisung AXLTKEY.

Die Anweisung AXLTKEY wird in der KTC-Umgebung gemäß ANSI X9.17 verwendet. Sie unterstützt die Verarbeitung der Serviceanforderungs-CSM (RFS-CSM) und die Generierung der entsprechenden Anforderungsantwort-CSM (RTR-CSM). Die RFS kann eine der folgenden Umwandlungsanforderungen enthalten:

- Einen Datenschlüssel KD1 umwandeln; für MAC dieser CSMS KD1 benutzen.
- Zwei Datenschlüssel KD1, KD2 umwandeln; für MAC dieser CSMS (KD1 XOR KD2) benutzen.
- Einen KK oder \*KK umwandeln; eingegebenen KDmac für MAC dieser CSM benutzen.

Wenn nur ein Datenschlüssel KD1 umgewandelt werden muß, wandelt AXLTKEY den Schlüssel um und gibt  $e*KM.C3(KD1)$  aus, das zur Überprüfung des MAC in der RFS und zur Generierung eines MAC für die betreffende RTR benutzt werden kann.

Müssen zwei Datenschlüssel KD1 und KD2 umgewandelt werden, wird die AXLTKEY-Funktion zweimal aufgerufen. Sie generiert  $e*KM.C3(KD1)$  und  $e*KM.C3(KD2)$  als Zwischenausgaben in den beiden Aufrufen. Das CFAP ruft dann ACOMBKD auf, um die beiden Ausgaben zu einem MAC-Schlüssel  $e*KM.CS(KD1 \text{ XOR } KD2)$  zu kombinieren, der für die MAC-Operation der RFS- und RTR-CSMS benutzt werden kann. Die von AXLTKEY generierten Zwischenausgaben können nur lokal ohne Exportberechtigung benutzt werden; dies wird durch die Steuervektorprüfung erzwungen.

Muß ein (\*)KK umgewandelt werden, so wandelt die AXLTKEY-Funktion den Schlüssel um und gibt  $e*KM.C3(KDmac)$  aus, das für die MACint-Operation der CSMS benutzt werden kann. KDmac begleitet den umzuwandelnden (\*)KK in der RFS; er wird unter dem (\*)KK mit Offset durch Null chiffriert. Es ist zu beachten, daß  $e*KM.C3(KDmac)$  nur lokal verwendet werden kann; die Exportkontrolle wird durch die Steuervektorprüfung in der Anweisung erzwungen.

# CC:

- 1. erfolgreiche Operation
- 2. C1L oder C1R ist ungültig

MA9-88-011

- 3. C2L oder C2R ist ungültig
- 4. C3 ist ungültig
- 5. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1L:

-- CV-Art	=	"KEK/ANSI"
-- AXLTKEY-Verwendungsbit	=	'1'
-- reserviert (48:63)	=	X'0'

- 2. Prüfung von C1R:

-- CV-Art	=	"KEK/ANSI"
-- AXLTKEY-Verwendungsbit	=	'1'
-- reserviert (48:63)	=	X'0'

- 3. Prüfung von C2L:

-- CV-Art	=	"KEK/ANSI"
-- AXLTKEY-Verwendungsbit	=	'1'
-- reserviert (48:63)	=	X'0'

- 4. Prüfung von C2R:

-- CV-Art	=	"KEK/ANSI"
-- AXLTKEY-Verwendungsbit	=	'1'
-- reserviert (48:63)	=	X'0'

- 5. Prüfung von C1L und C2L:

-- Schlüsselform (C1L)	=	Schlüsselform (C2L)
------------------------	---	---------------------

- 6. Prüfung von C1R und C2R:

-- Schlüsselform (C1R) = Schlüsselform (C2R)

- 7. Prüfung von C3:

-- Schlüsselart = "Datenschlüssel"

-- Exportkontrollbit 1 = 1 (kein Export)

--- In Fig. 60 sind die zulässigen Kombinationen von zu prüfenden C3-Attributen aufgeführt. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig. E, D, MG, MV, ACMB sind die Verwendungsbits für den Datenschlüssel-Steuervektor.

ANSI KDs kombinieren (ACOMBKD)

- Gleichung:  $e*KM.C1(KD1), e*KM.C2(KD2), C1, C2, C3$   
               $==== e*KM.C3(KD)$

- Eingabe:      $e*KM.C1(KD1)$      64 Bit langer ANSI-Datenschlüssel, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C1.  
               $e*KM.C2(KD2)$      64 Bit langer ANSI-Datenschlüssel, dreifach chiffriert unter dem Hauptschlüssel KM in Verbindung mit einem Steuervektor C2.  
                          C1     64-Bit-Steuervektor für den Datenschlüssel KD1.  
                          C2     64-Bit-Steuervektor für den Datenschlüssel KD2.  
                          C3     64-Bit-Steuervektor für den Ausgabe-MAC-Schlüssel KD.

- Ausgabe:      $e*KM.C3(KD)$      64 Bit langer ANSI-Datenschlüssel, dreifach chiffriert unter dem Hauptschlüssel in Verbindung mit dem Steuervektor C3.

Dieser Schlüssel kann zur Identifikationsüberprüfung einer Chiffrierservice-Meldung gemäß ANSI X9.17 mittels der Anweisungen GMAC und VMAC benutzt werden.

Beschreibung: In ANSI X9.17 werden Schlüssel ausgetauscht, indem die Paritäten mittels einer Folge von Chiffrierservice-Meldungen (CSMs) übermittelt werden. Bei jedem Schlüsselaustausch werden ein oder zwei Datenschlüssel übertragen. Die Integrität bestimmter CSMs wird geschützt, indem eine MAC-Operation mit der CSM und den Daten selber durchgeführt wird, oder beim Austausch von zwei Datenschlüsseln, indem die beiden KDs durch EXKLUSIV ODER verknüpft werden. Die Anweisung ACOMBKD betrifft den letzteren Fall, d.h. die Berechnung eines CSM MAC-Schlüssels aus zwei Datenschlüsseln.

ACOMBKD akzeptiert zwei 'partielle' MAC-Schlüssel, d.h., Datenschlüssel, die unter dem Hauptschlüssel in Verbindung mit einem Steuervektor chiffriert sind, welcher ihre Verwendung in ACOMBKD erlaubt. Partielle MAC-Schlüssel werden als optionales Nebenprodukt beim Import oder Export von ANSI-Datenschlüsseln durch ARTMK bzw. ARFMK erstellt.

ACOMBKD rekonstruiert die Datenschlüssel KD1 und KD2 intern und gibt (KD1 XOR KD2), chiffriert unter dem Hauptschlüssel in Verbindung mit einem Steuervektor, welcher die lokale Schlüsselbenutzung in den Anweisungen GMAC oder VMAC erlaubt, aus. Mit diesen Anweisungen können ankommende und abgehende CSMs einer Identifikationsüberprüfung unterzogen werden, wie sie vom CSM-Protokoll gemäß ANSI X9.17 gefordert wird.

HINWEIS: ACOMBKD muß überprüfen, daß die Eingabe-KDs nicht identisch oder äquivalent sind, und daß (KD1 XOR KD2) nicht gleich Null ist. Dadurch soll die Erstellung eines MAC-Schlüssels mit einem bekannten Wert (nämlich Null) verhindert werden.

Fig. 61 ist das Blockdiagramm für die Anweisung ACOMBKD.

# CC:

- 1. erfolgreiche Operation
- 2. C1 ist ungültig
- 3. C2 ist ungültig
- 4. C3 ist ungültig
- 5. fehlgeschlagene Operation (Fehler)

# Steuervektorprüfung:

- 1. Prüfung von C1:

- CV-Art = "Daten/ANSI"
  - ACMB-Verwendungsbit = '1'
  - reserviert (48:63) = X'0'

- In Fig. 62 sind die zulässigen Kombinationen von zu prüfenden C1-Attributen aufgeführt. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig. E, D, MG, MV, ACMB sind die Verwendungsbits für den Datenschlüssel-Steuervektor.

- 2. Prüfung von C2:

- CV-Art = "Daten/ANSI"
  - ACMB-Verwendungsbit = '1'
  - reserviert (48:63) = X'0'

- In Fig. 63 sind die zulässigen Kombinationen von zu prüfenden C2-Attributen aufgeführt. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig. E, D, MG, MV, ACMB sind die Verwendungsbits für den Datenschlüssel-Steuervektor.



- 3. Prüfung von C3:

-- Schlüsselart                   =   "Daten/ANSI"  
-- Exportkontrollbit 1           =   1 (kein Export)  
-- reserviert (48:63)           =   X'0'

--- In Fig. 64 sind die zulässigen Kombinationen von zu prüfenden C3-Attributen aufgeführt. Alle nicht in der Tabelle aufgeführten Kombinationen sind kryptographisch ungültig und deshalb nicht zulässig. E, D, MG, MV, ACMB sind die Verwendungsbits für den Datenschlüssel-Steuervektor.

ICV/OCV-Verwaltung

Ein einleitender Kettungswert (ICV) ist ein 64 Bit langer zufälliger, pseudozufälliger oder manchmal auch sich nicht wiederholender Wert, der in Verbindung mit dem Chiffreblockkettungs-Modus (CBC-Modus) des DEA und einigen Algorithmen zur Berechnung von Meldungs-Identifikationsüberprüfungs-codes (MACs) verwendet wird.

Bei der ICV-Verwaltung gibt es Optionen für die elektronische Übertragung und die lokale Speicherung sowohl von unchiffrierten als auch von chiffrierten ICVs. Die chiffrierten ICVs müssen allerdings zuerst dechiffriert werden, bevor sie als Eingabeparameter für eine der Chiffrierfunktionen verwendet werden können.

Ein Ausgabekettungswert (OCV) ist ein 64-Bit-Wert, der unter bestimmten Bedingungen von den Chiffrieranweisungen GMAC und VMAC ausgegeben wird. Die gleiche Chiffrieranweisung wird noch einmal aufgerufen, und der OCV wird als ICV übergeben. Der OCV ist immer in der Form eKM.CV(OCV) chiffriert, wobei CV ein Steuervektor für einen Zwischen-ICV ist. Bei der Anweisung VMAC ist ein chiffrierter OCV aus Sicherheitsgründen unbedingt erforder-

lich. Ein unchiffrierter OCV könnte in diesem Fall MACs offenlegen. Bei der Anweisung VMAC sollte das nicht möglich sein. Ein chiffrierter OCV ist auch für die Anweisung GMAC definiert. Dabei werden die Anweisungen GMAC und VMAC soweit wie möglich aneinander angeglichen, so daß eine Funktionsüberschneidung in der Hardware möglich wird.

#### ICV-Verwaltung außerhalb der Chiffriervorrichtung

Die Übertragungsarchitektur läßt die folgenden drei Modi der elektronischen Übertragung von ICVs zu:

- 1. Unchiffrierter ICV: wird im Klartext übertragen.
- 2. Chiffrierter ICV: wird mit einem Datenschlüssel (KD) chiffriert, den der Sender mit dem Empfänger teilt.
- 3. Privates Protokoll: der ICV wird mittels eines privaten Protokolls von Sender und Empfänger erstellt.

Unter der CA muß das CFAP sowohl unchiffrierte als auch chiffrierte ICVs verarbeiten. Anwendungen können aber auch ihre eigenen chiffrierten ICVs selber verwalten und dem CFAP unchiffrierte ICVs übergeben, um diese für die Übertragung zu chiffrieren und alle von anderen Knoten ankommenden chiffrierten ICVs zu dechiffrieren. Das Verschlüsselungsprogramm kann wahlweise auch ICVs mittels eines privaten Protokolls erstellen.

#### ICV-Verwaltung innerhalb der Chiffriervorrichtung

Für die elektronische Verteilung von ICVs werden keine Steuervektoren verwendet; der Steuervektor enthält auch keine Bits, die von der Chiffriervorrichtung (Hardware) zur Steuerung der ICV-Verteilung herangezogen werden können. Der ICV-Verteilungsmodus wird durch die Chiffriervorrichtung weder geprüft noch erzwungen. Die ICV-Verwaltung ist also strenggenommen eine Funk-

tion des Verschlüsselungsprogramms (d.h. der Software außerhalb der Chiffriervorrichtung).

Alle ICVs, die der Chiffriervorrichtung als Eingabeparameter für eine Chiffrierfunktion übergeben werden, müssen unchiffriert sein. ICV = 0, eine Forderung der Anweisungen GMAC und VMAC, ist nur ein Unterfall eines unchiffrierten ICV. Bei den betroffenen Chiffrierfunktionen handelt es sich um:

- 1. Chiffrierung
- 2. Dechiffrierung
- 3. MAC-Generierung
- 4. MAC-Überprüfung
- 5. Umwandlung von Chiffretext

#### Schlüsselverwaltung

Die CA-Schlüsselverwaltung umfaßt eine Reihe von Verfahren, Regeln und Prozeduren zur Verwaltung von Schlüsseln durch effektive Verwendung des Anweisungssatzes und anderer Bestandteile der Chiffriervorrichtung (z.B. Schlüsselladevorrichtung, Register usw.).

Der Chiffrierarchitektur liegt die Absicht zugrunde, daß die Schlüsselverwaltung mittels der angegebenen Chiffrierfunktionen auf die festgelegte Art und Weise durchgeführt wird. Dadurch wird sichergestellt, daß Systeme, in denen die gleiche Chiffrierarchitektur implementiert ist, nach dem gleichen Verfahren miteinander kommunizieren können. Im folgenden werden einige Merkmale der CA-Schlüsselverwaltung aufgeführt.

- Die Schlüsselverwaltung basiert auf dem Steuervektorkonzept, das neben anderen Vorteilen, ein mächtiges Instrument zur Erzwingung des Verwendungszwecks von Schlüsseln bietet, um

sicherzustellen, daß die Schlüssel getrennt und bestimmungsgemäß verwendet werden.

- In einem System intern folgt die Schlüsselverwaltung einem durch die Chiffriervorrichtung erzwungenen Zwei-Hierarchieebenen-Konzept: Nur der Hauptschlüssel wird unchiffriert innerhalb der physisch geschützten Chiffriervorrichtung gespeichert. Die anderen Schlüssel werden unter einem Chiffrierschlüssel, der durch EXKLUSIV-ODER-Verknüpfung des Hauptschlüssels mit einem dem Schlüssel zugeordneten Steuervektor gebildet wird, chiffriert und können außerhalb der Chiffriervorrichtung gespeichert werden. Außerhalb der Chiffriervorrichtung sind keine unchiffrierten Schlüssel zulässig.
- Hinsichtlich der Kommunikation existiert bei der Schlüsselverwaltung eine dreistufige Hierarchie, die vom CFAP (dem CF-Zugriffsprogramm) erzwungen wird: der Hauptschlüssel wird zum Chiffrieren von Schlüsselchiffrierschlüsseln benutzt, und Schlüsselchiffrierschlüssel, die die beteiligten Knoten gemeinsam besitzen, werden zum Chiffrieren anderer zwischen ihnen ausgetauschter Schlüssel verwendet.
- Sie umfaßt Prozeduren zur Initialisierung von Schlüsseln im System und in Netzwerken.
- Sie umfaßt Prozeduren zur Generierung, Verteilung, Übermittlung und Speicherung von Schlüsseln.
- Sie umfaßt Übertragungsprotokolle für die Übermittlung von Schlüsseln und Steuervektoren von einem Knoten an einen anderen.

#### Aufbauprinzip der CA-Schlüsselverwaltung

##### Steuervektorsystem

Ein Steuervektorsystem ist ein Chiffriersystem, in dem die CA implementiert ist.

#### Speicherung der Schlüssel und der Verschlüsselungsvariablen

Alle Schlüssel werden in der Form eKM.C(K) gespeichert, d.h. sie sind unter einem Schlüssel chiffriert, der durch EXKLUSIV-ODER-Verknüpfung des Hauptschlüssels mit einem Steuervektor gebildet wird. Zwischen-ICVs für MACs sowie Tokens werden ebenso chiffriert.

#### Schlüsselarten und Verschlüsselungsvariablenarten

Im CV-Art/Unterart-Feld im Steuervektor wird angegeben, welche Art von Schlüssel oder Verschlüsselungsvariablen chiffriert wird. In der CA sind dreizehn CV-Arten/Unterarten definiert:

- CV-Art/Unterart
  - Daten/Vertraulichkeit
  - Daten/MAC
  - Daten/XLT
  - Daten/Kompatibilität
  - Daten/ANSI
  - KEK/Sender
  - KEK/Empfänger
  - KEK/ANSI
  - PIN/Chiffrierung
  - PIN/Generierung
  - ICV/Zwischen-MAC
  - Schlüsselteil/
  - Token/

Die CV-Arten/Unterarten lassen sich in drei Kategorien untergliedern:

- 1. CA: definiert Schlüssel, die nur mit anderen CA-Systemen geteilt werden.
- 2. Kompatibilität: definiert Schlüssel, die mit anderen CA- oder Nicht-CA-Systemen geteilt werden.
- 3. ANSI: definiert Schlüssel, die gemäß dem ANSI X9.17-Schlüsselverteilungsprotokoll gesendet bzw. empfangen werden.

Kategorie	CV-Art/Unterart
CA	Daten/Vertraulichkeit
	Daten/MAC
	Daten/XLT
	KEK/Sender
	KEK/Empfänger
	KEK/ANSI
	PIN/Chiffrieren
	PIN/Generieren
	ICV/Zwischen-MAC
	Schlüsselteil/
	Token/
Kompatibilität	Daten/Kompatibilität
ANSI	Daten/ANSI
	KEK/ANSI

#### Schlüsselhierarchie

- 1. Hauptschlüssel: chiffriert alle Schlüssel, die lokal außerhalb der Chiffriervorrichtung gespeichert werden.
- 2. Schlüsselchiffrierschlüssel: chiffriert alle Schlüssel (außer dem Hauptschlüssel), die von einer Chiffriervorrichtung an eine andere übermittelt werden.
- 3. Datenschlüssel: chiffriert Daten und ICVs.
- 4. PIN-Chiffrierschlüssel: chiffriert PIN-Blöcke.

### Schlüsselverteilungsprotokolle

Die CA unterstützt drei Schlüsselverteilungsprotokolle:

- 1. Steuervektormodus (CV). Alle übertragenen Schlüssel haben die Form  $eKEK.C(K)$ .
- 2. Kompatibilitätsmodus (CV = 0). Alle übertragenen Schlüssel haben die Form  $eKEK(K)$ .
- 3. ANSI X9.17-Modus (ANSI). Alle übertragenen Schlüssel entsprechen dem ANSI-Standard X9.17.

### Schlüsselverteilungsregeln

Für die Verteilung (Export/Import) von Schlüsseln und Verschlüsselungsvariablen gilt ein Satz von Regeln, die die Schlüsselkategorien folgendermaßen zu den Verteilungsprotokollen in Beziehung setzen:

- 1. CA-Schlüssel und CA-Verschlüsselungsvariable müssen im Steuervektormodus exportiert bzw. importiert werden.
- 2. Kompatibilitätsschlüssel können entweder im Steuervektormodus (unter Verwendung des Steuervektors bei der Übertragung) oder im Kompatibilitätsmodus (mit CV = 0 bei der Übertragung) exportiert bzw. importiert werden.
- 3. ANSI-Schlüssel müssen im ANSI X9.17-Modus exportiert bzw. importiert werden.

Diese Regeln sind in der nachstehenden Tabelle zusammengefaßt:

Kategorie	CV-Art/Unterart	Schlüsselverteilungsprotokoll		
		CV	CV=0	ANSI
CA	Daten/Vertraulichkeit			
	Daten/MAC	j		
	Daten/XLT	j		
	KEK/Sender	j		
	KEK/Empfänger	j		
	KEK/ANSI	j		
	PIN/Chiffrieren	j		
	PIN/Generieren	j		
	ICV/Zwischen-MAC	j		
	Schlüsselteil/	j		
	Token/	j		
Kompatibilität	Daten/Kompatibilität	j	j	
ANSI	Daten/ANSI			j
	KEK/ANSI			j

Legende: "j" bedeutet, daß die Variable nach dem betreffenden Schlüsselverteilungsprotokoll verteilt werden kann.

#### Schlüsselstatus (nicht ANSI)

In der CA sind drei Schlüsselstatus definiert:

- 1. Lokal: Der Schlüssel wird unter KM chiffiert, d.h. in einer für die lokale Benutzung geeigneten Form.
- 2. Export: Der Schlüssel wird unter einem KEK/Sender chiffriert, d.h. in einer für den Export an ein anderes Gerät geeigneten Form.
- 3. Import: Der Schlüssel wird unter einem KEK/Empfänger chiffriert, d.h. in einer für den Import auf diesem Gerät geeigneten Form.



### Schlüsselgenerierung

- 1. Jedem von einem Steuervektorsystem generierten Schlüssel ist ein Steuervektor zugeordnet.
- 2. Schlüssel können je nach Art/Unterart des zur Chiffrierung des generierten Schlüssels verwendeten Steuervektors a) im lokalen Status, b) im Exportstatus oder c) im Importstatus generiert werden (Punkt G in dieser Liste).
- 3. Schlüssel werden mit den Anweisungen GKS und KGEN generiert.

GKS generiert einen Schlüssel in zwei Formen unter Verwendung von zwei Steuervektoren. Die CV-Art/Unterart muß bei beiden zu der CA-Kategorie gehören. (Ein Schlüssel kann nicht gleichzeitig einer CA-Kategorie und einer Kompatibilitätskategorie angehören.) KGEN generiert einen Schlüssel in einer einzigen Form unter Verwendung eines Steuervektors der CA-Kategorie oder der Kompatibilitätskategorie.

### Schlüsselstatusumwandlungen (nicht ANSI)

Folgende Schlüsselstatusumwandlungen (durch die angegebenen Anweisungen) werden von der CA unterstützt:

Eingabestatus	Ausgabestatus		
	Lokal	Import	Export
Lokal	-	-	RFMK
Import	RTMK	-	XLTKY
Export	-	-	-

Legende: "-" bedeutet, daß die betreffende Umwandlung nicht unterstützt wird.

### Anweisungsfolgen für die Schlüsselverwaltung

In den Tabellen Fig. 65, 66, 67 und 68 sind die CA-Funktionen aufgeführt, die für die Schlüsselverwaltung verwendet werden können. Ausführliche Beschreibungen sind den nachfolgenden Abschnitten zu entnehmen.

### Schlüsselstatus und Anweisungen

In Fig. 69 ist die Beziehung zwischen lokalen, exportierten und importierten Schlüsseln und die Möglichkeiten zu ihrer Erstellung oder Umwandlung durch verschiedene Anweisungen in der Chiffriervorrichtung aufgeführt.

Ein lokaler Schlüssel wird im Schlüsselspeicher gespeichert und kann als Eingabeparameter für Chiffrieranweisungen verwendet werden. Er hat die Form  $eKM.C(K)$ , wobei KM der in der Chiffriervorrichtung gespeicherte Hauptschlüssel und C der dem Schlüssel K zugeordnete Steuervektor ist. Ein Exportschlüssel ist ein lokaler Schlüssel, der einen Kanal für die Übermittlung von Schlüsseln an ein anderes System definiert. Ein Importschlüssel ist ein lokaler Schlüssel, der einen Kanal für den Empfang von Schlüsseln von einem anderen oder dem eigenen System definiert. Ein exportierter Schlüssel ist ein Schlüssel der Form  $eKEK.C(K)$ , wobei KEK ein KEK/Sender und C der dem Schlüssel K zugeordnete Steuervektor ist. Ein importierter Schlüssel ist ein Schlüssel der Form  $eKEK.C(K)$ , wobei KEK ein KEK/Empfänger und C der dem Schlüssel K zugeordnete Steuervektor ist.

RFMK wandelt einen lokalen Schlüssel in einen exportierten Schlüssel um. RTMK wandelt einen importierten Schlüssel in einen lokalen Schlüssel um. KGEN generiert eine Form eines lokalen Schlüssels. XLTKEY wandelt einen importierten Schlüssel in einen exportierten Schlüssel um. GKS generiert einen Schlüssel in der Regel in zwei Formen, die unterschiedliche Verwendungsattribute

aufweisen können. GKS OP-OP generiert einen Schlüssel in den zwei Formen "lokal" und "exportiert". GKS OP-IM generiert einen Schlüssel in den zwei Formen "lokal" und "importiert". GKS IM-EX generiert einen Schlüssel in den zwei Formen "importiert" und "exportiert". GKS EX-EX generiert einen Schlüssel in den zwei Formen "exportiert" und "exportiert".

### Übersicht über die Schlüsselarten

Die CA-Anweisung arbeitet mit folgenden Schlüsselarten:

- 1. Schlüsselchiffrierschlüssel

Diese Schlüssel werden zum Chiffrieren anderer Schlüssel verwendet. Der Hauptschlüssel, Mehrdomänenschlüssel und Terminalhauptschlüssel sind Schlüsselchiffrierschlüssel. Alle Schlüsselchiffrierschlüssel mit Ausnahme des Hauptschlüssels können in folgende Unterarten eingeteilt werden:

- Schlüsselchiffrierschlüssel Sender (KEK/Sender): ein Schlüssel, der benötigt wird, um Schlüssel an andere Knoten zu senden.
- Schlüsselchiffrierschlüssel Empfänger (KEK/Empfänger): ein Schlüssel, der benötigt wird, um Schlüssel von anderen Knoten zu empfangen.
- ANSI-Schlüsselchiffrierschlüssel (KEK/ANSI): ein Schlüssel, der in der Schlüsselverwaltungsumgebung gemäß ANSI X9.17 benutzt wird. Der ANSI KEK ist nicht richtungsgebunden wie ein regulärer CA-KEK (d.h. KEK/Sender bzw. KEK/Empfänger).
- Hauptschlüssel - Der Hauptschlüssel dient zum Schutz aller anderen Schlüssel im System.

Nur der Hauptschlüssel muß unchiffriert innerhalb der geschützten Chiffriervorrichtung gespeichert werden. Die anderen Schlüssel sind mittels Chiffrierung durch EXKLUSIV-ODER-Verknüpfung des Hauptschlüssels mit einem dem Schlüssel zugeordneten Steuervektor geschützt. Da sie chiffriert sind, können sie in Speicherbereichen außerhalb der Chiffriervorrichtung gespeichert werden.

Die CA fordert, daß der Hauptschlüssel doppelte Länge, d.h. 128 Bit, besitzt, von denen 112 Bit den eigentlichen Wert darstellen und 16 Bit Paritätsbits sind.

- Mehrdomänenschlüssel - Die Schlüsselverwaltung verlangt einen eindeutigen Schlüssel doppelter Länge mit Parität (jeweils 128 Bit), der mit jedem CV-Knoten geteilt wird, mit dem kommuniziert werden soll.

Dies gilt auch für Nicht-CV-Systemknoten, wo bei Systemen mit Schlüsseln einfacher Länge ein Schlüssel doppelter Länge so gespeichert wird, daß die linke und die rechte Hälfte identisch sind. Die Mehrdomänenschlüssel werden zum Chiffrieren (oder Versenden) und Abrufen von Schlüsseln benutzt, die zwischen Knoten ausgetauscht werden, welche die Mehrdomänenschlüssel gemeinsam besitzen.

- Terminalhauptschlüssel - Die Schlüsselverwaltung verlangt einen eindeutigen Terminalhauptschlüssel, den alle Terminals besitzen, mit denen kommuniziert werden soll.

In Terminals, bei denen Schlüssel nur gesendet, aber nicht empfangen werden müssen, wird gewöhnlich nur ein Terminalhauptschlüssel einfacher Länge (64 Bit) gespeichert. In diesen Fällen wird der Schlüssel doppelter Länge im Schlüsselspeicher gespeichert, wobei die rechte und die linke Schlüsselhälfte identisch sind.

In den Fällen, in denen im Terminal ein Schlüssel doppelter Länge gespeichert wird, wird dieser unverändert im Schlüsselspeicher gespeichert.

- 2. **Datenschlüssel** - Datenschlüssel werden zum Chiffrieren von Daten benutzt. Identifikationsüberprüfungsschlüssel, Dateischlüssel und Sitzungsschlüssel gehören in diese Kategorie.
- 3. **PIN-Schlüssel** - PIN-Schlüssel werden in zwei Unterarten eingeteilt:
  - PIN-Chiffrierschlüssel: diese Schlüssel werden zum Chiffrieren von PINs benutzt.
  - PIN-Generierungsschlüssel: diese Schlüssel werden im PIN-Generierungsalgorithmen zur Generierung von PINs benutzt.
- 4. **Schlüsselteil** - Ein Schlüsselteil ist ein Teil oder eine Komponente eines Schlüssels, besitzt aber die gleiche Länge wie ein Schlüssel. Beispiel: ein Schlüssel K kann aus zwei Schlüsselteilen  $K_a$  und  $K_b$  bestehen, für die gilt  $K_a \text{ XOR } K_b = K$ .
- 5. **Zwischen-ICV** - Ein Zwischen-ICV wird zum Chiffrieren des Zwischen-Ausgabekettungsvektors eines Daten- oder Meldungssegments benutzt.

Dieser OCV wird dann als ICV an das nächste Datensegment übergeben. Dies geschieht, wenn eine Meldung oder ein Datensegment, für die bzw. das ein MAC generiert oder überprüft werden soll, zu lang ist und in kürzere Segmente unterteilt werden muß.

- 6. **Token** - Tokens sind Variable zum Schutz der Integrität der in der Datenschlüsseldatei (einem Schlüsselspeicher für Datenschlüssel) gespeicherten Datenschlüssel.

Die tragen dazu bei, den Zugriff nicht berechtigter Benutzer auf Datenschlüssel zu verhindern.

### Schlüsselinitialisierung

Im folgenden werden die Schlüsselinitialisierungsmechanismen beschrieben.

Bei der Systeminstallation muß der Hauptschlüssel als erster Schlüssel installiert werden. Anschließend können einige Schlüsselchiffrierschlüssel wie z.B. Mehrdomänenschlüssel oder Transportschlüssel manuell installiert werden. Sobald die Mehrdomänenschlüssel verfügbar sind, kann das System Schlüssel und chiffrierte Daten mit anderen Systemen austauschen.

Der Hauptschlüssel und andere Schlüsselchiffrierschlüssel müssen unter geeigneten Zugangskontrollen manuell installiert werden.

Die Schlüsseleingabevorrichtung, über die der Schlüssel in die Chiffriervorrichtung eingegeben wird, muß eine sichere Schnittstelle zur Chiffriervorrichtung besitzen wie z.B. die Anzeige (falls sie nicht Bestandteil der Schlüsseleingabevorrichtung ist). Darauf wird im Abschnitt über physische Einrichtungen und Schnittstellen eingegangen.

Wenn ein Hauptschlüssel oder ein anderer Schlüsselchiffrierschlüssel manuell installiert werden muß, ist eine Doppeleingabe wünschenswert, die darauf abzielt, keiner Einzelperson die vollständige Kenntnis der Schlüssel zu ermöglichen. Bei der Doppeleingabe wird der Schlüssel in zwei Teilen eingegeben. Alle Schlüsselteile haben die gleiche Länge und werden von verschiedenen Personen eingegeben. Der tatsächliche Schlüssel wird durch EXKLUSIV-ODER-Verknüpfung der beiden Schlüsselteile gebildet.

Die CA unterstützt auch eine Mehrfach-Schlüsseleingabe (d.h., wenn ein Schlüssel mehr als zwei Schlüsselteile besitzt; der Schlüssel kann z.B. n Schlüsselteile KP1, KP2,... KPn besitzen, wobei  $K = KP1 \text{ XOR } KP2 \dots \text{ XOR } KPn$  ist).

Wenn bei einem System eine Doppeleingabe erforderlich ist, müssen die Schlüsselladevorrichtung und andere Einrichtungen der Chiffriervorrichtung so gestaltet sein, daß sie dies unterstützen. Es sollten beispielsweise zwei physische Schlösser (oder ein Schloß, bei dem für zwei Positionen zwei verschiedene Schlüssel benötigt werden) vorhanden sind, damit jeder Schlüsselteil separat eingegeben werden kann. In Systemen ohne physische Schlüssel können Paßwörter verwendet werden, wobei für jeden Schlüsselteil ein anderes Paßwort eingegeben werden muß. Bei der Eingabe eines Schlüssels oder eines Schlüsselteils darf die Eingabe nur durch die eingebende Person visuell überprüft werden können. Wenn diese Person festgestellt hat, daß die Anzeige mit der Eingabe übereinstimmt, kann sie das Laden des Schlüssels oder Schlüsselteils in Register innerhalb der Chiffriervorrichtung aktivieren (z.B. mittels einer Taste, die die Schnittstelle zwischen der Schlüsseleingabevorrichtung und den Schlüsselregistern bildet). Jeder einzugebende Schlüssel oder Schlüsselteil muß doppelte Länge einschließlich ungerader 16-Bit-Parität aufweisen. Die Parität dient der Fehlererkennung.

#### Manuelle Eingabe des Hauptschlüssels

Die CA gibt kein spezielles Verfahren zur manuellen Installation des Hauptschlüssels vor. Das im vorliegenden Abschnitt skizzierte Verfahren ist als Richtlinie aufzufassen. Es wird empfohlen, den eingegebenen Hauptschlüssel temporär in einem Register für den neuen Hauptschlüssel zu speichern und erst dann zu aktivieren, wenn die Funktion "Hauptschlüssel setzen" aufgerufen wird.

Bei dieser Strategie kann der eingegebene Hauptschlüssel überprüft und gegebenenfalls neu eingegeben werden, und der eingegebene Hauptschlüssel muß nicht aktiviert werden, solange nicht feststeht, daß er korrekt ist und alle Verarbeitungsanforderungen wie z.B. die Umchiffrierung anderer Schlüssel vom aktuellen Hauptschlüssel unter den neuen Hauptschlüssel erfüllt sind.

Im Rahmen der manuellen Eingabe des Hauptschlüssels muß an der physischen Schnittstelle und/oder an der Programmierschnittstelle ein 32 oder 64 Bit langes nicht geheimes Prüfmuster (Funktionieren des eingegebenen Hauptschlüssels) ausgegeben werden. Alternativ kann das Prüfmuster als Antwort auf den Aufruf einer Anweisung durch ein Schlüsselinstallations-Dienstprogramm erzeugt werden. Das Prüfmuster ist in Umgebungen nützlich, in denen die Sicherheit nicht gewährleistet ist, wenn ein eingegebener Schlüssel zur Überprüfung durch den Benutzer auf einem Bildschirm oder einer Konsole angezeigt wird. Spezielle Prüfverfahren sind nicht Gegenstand dieses Dokuments.

Die manuelle Schlüsseleingabe (des Hauptschlüssels oder der Schlüsselchiffrierschlüssel) kann implementiert werden, indem eine Tastatur über eine physisch sichere Schnittstelle an die CF angeschlossen wird. Um die Funktion zum Laden der über die Tastatur eingegebenen Schlüssel zu aktivieren oder deaktivieren, kann ein physischer Schlüsselschalter und/oder eine Paßwortkontrolle verwendet werden.

#### Aktivierung des Hauptschlüssels

Ein eingegebener Hauptschlüssel wird aktiviert, indem die Funktion "Hauptschlüssel setzen" aufgerufen wird. Dies bewirkt, daß der Inhalt des Registers, in dem der neue Hauptschlüssel gespeichert ist, in das Hauptschlüsselregister (d.h. das Register für den aktuellen Hauptschlüssel) geladen wird.



Die CA ermöglicht wahlweise die Verwendung eines Registers für den alten Hauptschlüssel. In diesem Fall bewirkt die Funktion "Hauptschlüssel setzen" zuerst, daß der Inhalt des Hauptschlüsselregisters in das Register für den alten Hauptschlüssel geladen wird, und dann, daß der Inhalt des Registers für den neuen Hauptschlüssel in das Hauptschlüsselregister geladen wird.

Jedem Hauptschlüsselregister (für den neuen, den aktuellen und wahlweise für den alten Hauptschlüssel) ist eine Markierung zugewiesen, das gesetzt wird (= 1), wenn ein Schlüssel aktiv in dem Register gespeichert wurde, und das zurückgesetzt wird (= 0), sobald ein Schlüssel deaktiviert oder die Chiffriervorrichtung nach dem Abschalten wieder eingeschaltet wird.

#### Schlüsselinstallation über die Programmschnittstelle

Schlüssel können auch über die Programmschnittstelle installiert werden. Dazu ist jedoch eine manuelle Intervention durch einen Sicherheitsbeauftragten (oder Urheber), der über den physischen Schlüssel oder die für den Zugriff erforderlichen Paßwörter verfügt, erforderlich. Bei diesem Verfahren ändert der Benutzer den Arbeitsstatus der Chiffriervorrichtung mit Hilfe des physischen Schlüssels (oder Paßworts) in einen Status, der als Extrasicher-Modus bezeichnet wird, und gibt dann einen unchiffrierten Schlüssel und den zugehörigen Steuervektor in die Anweisung EMK ein, die nur in diesem Modus ausgeführt werden kann. Die Funktion EMK chiffriert den Klartextschlüssel unter einem Schlüssel, der durch EXKLUSIV-ODER-Verknüpfung des Hauptschlüssels mit dem Steuervektor gebildet wird. Der chiffrierte Schlüssel und der Steuervektor werden zur Verwendung im System gespeichert.

#### Vorgeschlagene Regeln für Setzen, Prüfen und Erzwingen ungerader Parität

Jeder kurze Schlüssel besteht aus 64 Bit, die konventionsgemäß von 0 bis 63 durchnumeriert werden, wobei Bit 0 das höchstwertige Bit ist. Die Paritätsbits sind Bit 7, Bit 15, Bit 23, Bit 31, Bit 39, Bit 47, Bit 55 und Bit 63, wobei Bit 7 das Paritätsbit für die Schlüsselbits 0 bis 6 ist und Bit 15 das Paritätsbit für die Schlüsselbits 8 bis 14 usw.

#### Setzen ungerader Schlüsselparität

Alle in das System importierten Nicht-Datenschlüssel müssen ungerade Parität haben. Dazu zählen der Hauptschlüssel und alle chiffrierten und im Schlüsselspeicher gespeicherten Schlüsselchiffrierschlüssel, PIN-Chiffrierschlüssel und PIN-Generierungsschlüssel.

In allen generierten Schlüssel wird ungerade Parität gesetzt, außer wenn es sich um einen Schlüssel handelt, der als Zufallszahl generiert und als bereits unter einem anderen Schlüssel chiffrierter gewünschter Schlüssel definiert wird.

#### Prüfen der Schlüsselparität

Bevor die Chiffriervorrichtung einen Schlüssel benutzt, muß sie prüfen, ob der Schlüssel ungerade Parität besitzt. Es sollten BedingungsCodes gesetzt werden, die angeben, ob der Schlüssel ungerade Parität besitzt oder nicht. Das CFAP muß entscheiden, ob das Nichtvorhandensein ungerader Parität ein Fehler ist, und ob die Ausgabe einer angeforderten Funktion mit dem betreffenden Schlüssel vertrauenswürdig ist oder nicht.

#### Erzwingen ungerader Schlüsselparität durch die Hardware

Die Hardware muß beim Hauptschlüssel ungerade Parität erzwingen. Wird ein Paritätsfehler festgestellt, muß die angeforderte Chiffrieranweisung abgebrochen werden. Die Hardware sollte einen

Wiederherstellungsversuch unternehmen (z.B. mit einer Sicherungskopie des Hauptschlüssels) und einen Bedingungscode setzen, der darauf hinweist, daß ein nicht behebbarer Paritätsfehler im Hauptschlüssel aufgetreten ist.

In bestimmten ausgewählten Fällen kann ein Paritätsfehler bei einem Schlüsselchiffrierschlüssel oder einem anderen Nicht-Datenschlüssel bewirken, daß eine angeforderte Chiffrierfunktion abgebrochen wird.

#### Schlüsselgenerierung

Schlüssel sollten zufallsmäßig generiert werden. Dies impliziert einen Zufallszahlengenerator in Form einer speziellen Hardware, oder es wird eine Software-Implementierung eines akzeptablen Algorithmus zur elektronischen Generierung von Schlüsseln benötigt. Schlüssel können auch manuell in zufallsmäßiger Weise generiert werden, z.B. durch Werfen einer Münze. Die Schlüssel können bei der Generierung mit einer Parität versehen oder ohne Parität erzeugt werden. Wenn eine Parität gewünscht wird, müssen die Schlüssel ungerade Parität besitzen.

#### Manuelle Schlüsselgenerierung

Die CA empfiehlt, Hauptschlüssel und Schlüsselchiffrierschlüssel manuell zu installieren und zu generieren. Ein geeignetes Verfahren für die manuelle Generierung von Schlüsseln ist das Werfen einer Münze oder das Würfeln. Eine Möglichkeit, Schlüssel durch Würfeln zu erzeugen, könnte beispielsweise folgendermaßen aussehen:

Der zuständige Kurier oder Sicherheitsbeauftragte wählt acht 16-seitige Würfel ohne Vorzugsseite aus und ordnet jeder Würfelseite eine Zahl von 0 bis 15 als Hexadezimalwert zu. Angenommen,

Seite 1 hat den Wert 0, dann hat Seite 16 den Wert F usw. Dann werden folgende Schritte ausgeführt:

- 1. Würfeln (auf einer festen, ebenen Oberfläche) und die Ergebnisse in 8 Hexadezimalziffern notieren.
- 2. Die Hexadezimalwerte in acht Bytes mit jeweils 8 Binärstellen umschreiben.
- 3. Die ungerade Parität der ersten sieben Bits jedes Byte berechnen. Für 8 Byte werden insgesamt 8 Paritätsbits gebildet.
- 4. Das achte (letzte) Bit jedes Byte durch das im vorhergehenden Schritt berechnete Paritätsbit ersetzen. Die Binärwerte in 8 Hexadezimalziffern umschreiben. Das Ergebnis ist dann der manuell generierte Schlüssel in Hexadezimaldarstellung.

#### Elektronische Schlüsselgenerierung

Schlüssel können mit den im Anweisungssatz-Abschnitt beschriebenen Anweisungen KGEN und GKS generiert werden.

F

- Generierung durch KGEN

Die Anweisung KGEN generiert Schlüssel, die unmittelbar im System benutzt werden können.

- KGEN erlaubt die Generierung unchiffrierter Schlüssel. Dieser Modus wird benutzt, wenn Kuriere eingesetzt werden.
- KGEN erlaubt die Generierung von Schlüsseln, die unter einer EXKLUSIV-ODER-Verknüpfung des Hauptschlüssels mit

einem speziellen Steuervektor chiffriert sind. Dieser Steuervektor wird der Anweisung vom CFAP übergeben und wird von der Anweisung auf gültige Kombinationen überprüft.

- Durch GKS

Die Anweisung GKS generiert Schlüssel und gibt sie in chiffrierter Form aus. Bei dieser Anweisung werden die Schlüssel immer paarweise erzeugt, d.h. zwei Exemplare eines Schlüssels, in folgenden Modi:

-- Modus 0: OP-OP (lokal-lokal)

Dieser Modus generiert nur Datenschlüssel für die lokale Verwendung im System. Es werden zwei Exemplare eines Datenschlüssels generiert. Jedem Schlüsselexemplar ist ein Steuervektor mit anderen Attributen zugeordnet. Ein Exemplar des Schlüssels hat beispielsweise das Attribut MG (GMAC), das andere das Attribut MV (VMAC). Beide Exemplare werden unter einer EXKLUSIV-ODER-Verknüpfung des Hauptschlüssels und des entsprechenden Steuervektors chiffriert.

**F** -- Modus 1: OP-EX (lokal - Export)

Dieser Modus generiert ein Schlüsselexemplar zur lokalen Verwendung (und Speicherung) und ein Schlüsselexemplar in einer Form, die an einen anderen CV-Knoten exportiert werden kann. Das erste Exemplar, das hier als lokales Exemplar bezeichnet wird, wird unter dem Hauptschlüssel (nach EXKLUSIV-ODER-Verknüpfung mit dem zugeordneten Steuervektor) chiffriert und kann später, abhängig von der Angabe im Exportkontrollfeld an andere Knoten exportiert werden (mittels RFMK). (Es wird angenommen, daß der Eigner des Schlüssels zum Zeitpunkt der Generierung weiß, ob der generierte Schlüssel für den Export zugelassen sein soll.) Das andere Exem-

plar, hier als Exportexemplar bezeichnet, wird unter einem KEK/-Sender des generierenden Knotens (nach EXKLUSIV-ODER-Verknüpfung mit dem zugeordneten Steuervektor) chiffriert.

-- Modus 2: EX-EX (Export - Export)

Ein in diesem Modus generierter Schlüssel kann nicht lokal im System benutzt werden. Dieser Modus generiert zwei Schlüssel-exemplare in Formen, in denen sie an zwei andere CV-Knoten exportiert werden können. Jedes Exemplar des generierten Schlüssels, hier als Exportexemplar bezeichnet, wird unter einem anderen KEK/Sender (nach EXKLUSIV-ODER-Verknüpfung mit dem zugeordneten Steuervektor) chiffriert. Den beiden Exemplaren sind Steuervektoren mit verschiedenen Verwendungsattributen zugeordnet.

Dieser Modus ist nützlich, wenn Schlüssel von einem Knoten aus versendet werden, der als Verteilungszentrum dient.

-- Modus 3: OP-IM (lokal - Import)

Dieser Modus generiert ein Schlüsselexemplar für die lokale Verwendung und ein Schlüsselexemplar in einer Form, in der es vom generierenden Knoten importiert werden kann. Das erste Exemplar (das sogenannte lokale Exemplar) wird unter dem Hauptschlüssel (nach EXKLUSIV-ODER-Verknüpfung mit dem zugeordneten Steuervektor) chiffriert und kann später, abhängig von der Angabe im Exportkontrollfeld, an andere Knoten exportiert werden (mittels RFMK). Das andere Exemplar (das sogenannte Importexemplar) wird unter einem KEK/Empfänger (nach EXKLUSIV-ODER-Verknüpfung mit dem zugeordneten Steuervektor) chiffriert.

Dieser Modus ist für Dateianwendungen nützlich. Bei der Archivierung sensibler Daten werden die chiffrierten Daten und die Datenschlüssel beispielsweise (in chiffrierter Form) auf Band gespeichert. Aber unter welchem Schlüssel muß der auf dem Band

gespeicherte Datenschlüssel chiffriert werden? Es ist nicht günstig, den Datenschlüssel unter dem Hauptschlüssel chiffriert zu speichern, da der Hauptschlüssel später geändert werden kann. Praktischer ist es, den Datenschlüssel unter einem KEK/Sender (der in diesem Fall der Hauptschlüssel für die Datei ist) zu chiffrieren, da KEKs eine längere Lebensdauer haben. In einer Anwendung generiert die GKS im OP-IM-Modus ein Datenschlüssel-paar. Das lokale Exemplar hat die CV-Art = Daten/Vertraulichkeit und das Verwendungsattribut für Chiffrierung. Das Importexemplar hat die CV-Art = Daten/XLT und das Attribut XDin = 1. Die Daten werden dann durch das lokale Exemplar chiffriert (mit der Anweisung "Chiffrieren"). Die chiffrierten Daten und das Importexemplar werden dann archiviert. Zu einem späteren Zeitpunkt, wenn die Daten abgerufen und unter einen neuen Schlüssel umchiffriert werden sollen, wird das Importexemplar vom Band geladen und rekonstruiert (mit der Anweisung RTMK). Da das Importexemplar das Attribut XDin = 1 besitzt, kann es in der Anweisung "Chiffretext umwandeln" zur Umchiffrierung der Daten unter einen neuen Schlüssel benutzt werden. Andere identifizierte Anwendungen sind solche, bei denen das lokale Exemplar und das Importexemplar die in Fig. 37 angegebenen Verwendungsattribute besitzen.

HINWEIS: In diesem Modus können nur Datenschlüssel erzeugt werden.

#### -- Modus 4: IM-EX (Import - Export)

Dieser Modus generiert ein Schlüsselexemplar für den späteren Import durch den generierenden Knoten und ein Schlüsselexemplar in einer Form, in der es an einen anderen Knoten exportiert werden kann. Das erste Exemplar (das sogenannte Importexemplar) wird unter einem KEK/Empfänger (nach EXKLUSIV-ODER-Verknüpfung mit dem zugeordneten Steuervektor) chiffriert. Es kann später vom generierenden Knoten mittels der Anweisung RTMK abgerufen werden. Das andere Exemplar (das sogenannte Exportexemplar) wird

unter einem KEK/Sender (nach EXKLUSIV-ODER-Verknüpfung mit den zugeordneten Steuervektor) chiffriert.

Dieser Modus ist bei IBM SNA-Mehrdomänenanwendungen nützlich, wo z.B. Sitzungsschlüssel generiert und verteilt werden.

HINWEIS: Für jedes Exemplar des generierten Schlüssels muß das CFAP den zugeordneten Steuervektor an die Anweisung GKS übergeben. Die Anweisung GKS prüft die Steuervektoren der generierten Schlüsselexemplare je nach Anweisungsmodus auf gültige Verwendungsattribute und gültige Kombinationen anderer Attribute.

#### Schlüsselverteilung

Schlüssel können manuell (z.B. durch Kuriere) oder elektronisch verteilt werden. Die Verteilung durch Kuriere ist in größeren Netzwerken mit zahlreichen Knoten aus Kostengründen nicht wünschenswert. In vielen Fällen wird deshalb die elektronische Schlüsselverteilung bevorzugt.

#### Schlüsselverteilungsprotokolle

Verschlüsselungsknoten in einem Netzwerk sind gewöhnlich eine Mischung von Knoten mit Steuervektor-Verarbeitungsmöglichkeit und Knoten ohne Steuervektor-Verarbeitungsmöglichkeit. Die erste Art von Knoten wird als CV-Knoten bezeichnet, die zweite als Nicht-CV-Knoten. Beispiele für Nicht-CV-Knoten sind einige vorhandene IBM Verschlüsselungsprodukte wie IBM 4700 oder IBM 3848/CUSP. Für die Schlüsselverteilung unter Knoten der gleichen oder unterschiedlicher Art gibt es folgende Schlüsselverteilungsprotokolle:

- Steuervektormodus (CV)
- Kompatibilitätsmodus (CV = 0)
- ANSI X9.17-Modus (ANSI)



Der Steuervektormodus ist ein Verfahren, bei dem alle von einer Chiffriervorrichtung an eine andere übertragenen Schlüssel nach Schlüsselart und Verwendungszweck des Schlüssels kryptographisch getrennt sind. Alle übermittelten Schlüssel haben die Form  $eKEK.C(K)$ , wobei K der zu übertragende Schlüssel, KEK der Chiffrierschlüssel, unter dem K chiffriert ist, und C der K zugeordnete Steuervektor ist. Der Steuervektor C kann mit dem chiffrierten Schlüssel  $eKEK.C(K)$  zusammen übertragen werden, muß aber nicht. Dieser Modus wird als CV-Modus bezeichnet, weil bei der Übertragung Steuervektoren benutzt werden.

Der Kompatibilitätsmodus ist ein Verfahren, bei dem alle von einer Chiffriervorrichtung an eine andere übertragenen Schlüssel die Form  $eKEK(K)$  besitzen. Es erfolgt keine kryptographische Schlüsseltrennung. Im Kompatibilitätsmodus können Schlüssel an vorhandene IBM Chiffriersysteme (PCF, CUSP/3848, 4700) verteilt werden, die mit diesem Schlüsselverteilungsverfahren arbeiten. Derzeit beschränkt sich das SNA-Verschlüsselungsprogramm auf die Schlüsselverteilung nach diesem Verfahren. Die Schlüsselverteilung in diesem Modus ist auf Datenschlüssel (mit Verwendungsattributen für Chiffrierung und Dechiffrierung) sowie MAC-Schlüssel (mit MG- und MV-Verwendungsattributen) beschränkt. Dieser Modus wird mit CV = 0 bezeichnet, da er mit der Verwendung eines als lauter Nullen bestehenden Steuervektors gleichbedeutend ist. Die Schlüsselverteilung erfolgt mit CV = 0 bei der Übertragung. Die erfolgt ferner mit CV = 0 unter den CA-Chiffrieranweisungen und den CA CFAP-Makros.

Der ANSI X9.17-Modus ist ein Schlüsselverteilungsverfahren, das den ANSI-spezifischen Chiffrieranweisungen der CA und einer Untergruppe von ANSI-spezifischen CFAP-Makros der DDA entspricht.

Jeder Schlüsselverteilungsmodus besitzt eine Gruppe von Regeln und Prozeduren, die zusammen das Schlüsselverteilungsverfahren definieren. Jeder Schlüsselchiffrierschlüssel, den Chiffrierein-

richtungen zum Zweck der Schlüsselübertragung untereinander benutzen, definiert einen kryptographischen Übertragungskanal oder Schlüsselverteilungskanal. Dies kann ein CV-Kanal, ein CV=0-Kanal oder ein ANSI-Kanal sein, je nachdem, welches Schlüsselverteilungsprotokoll die beiden von dem betreffenden Kanal unterstützten Chiffriervorrichtungen verwenden. Ein CV-Kanal bedeutet daher, daß über diesen Kanal übertragene Schlüssel dem Steuervektormodus der Schlüsselverteilung entsprechen; ein CV=0-Kanal bedeutet, daß die über diesen Kanal übertragenen Schlüssel dem Kompatibilitätsmodus der Schlüsselverteilung entsprechen, und ein ANSI-Kanal bedeutet, daß die über diesen Kanal übertragenen Daten dem ANSI X9.17-Schlüsselverteilungsmodus entsprechen. Es ist auch sinnvoll, den gleichen Schlüsselchiffrierschlüssel alternativ für zwei Kanäle zu definieren (z.B. als CV-Kanal und als CV=0-Kanal, je nachdem, welcher Modus von der Anwendung ausgewählt und sich letztendlich als Parameter auf der Ebene der Chiffrieranweisungen niederschlägt).

Der Schlüsselverteilungskanal ist ein bequemes Konzept, besonders weil das Schlüsselverteilungsverfahren nicht notwendigerweise das in den Chiffriervorrichtungen der betreffenden Chiffriersysteme verwendete Schlüsselverwaltungsverfahren widerspiegelt oder von diesem abhängt. Die Schlüsselverwaltung kann beispielsweise auf Steuervektorbasis oder auf Variantenbasis oder auf sonstige Weise durchgeführt werden, wohingegen das für die Übertragung von Schlüsseln ausgewählte Schlüsselverteilungsverfahren von Fragen wie dem Kommunikationspartner, der Einhaltung von Standards und Praktiken, früheren Netzwerkvereinbarungen usw. abhängt.

Eine Chiffriervorrichtung gemäß der CA (ein sogenannter CA-Knoten) kann einen CV-Kanal für die Kommunikation mit einem anderen CA-Knoten errichten; sie kann einen CV=0-Kanal für die Kommunikation mit einem IBM PCF-, IBM CUSP/3848- oder IBM 4700-Systemknoten errichten; oder sie kann einen ANSI-Kanal für die Kommu-

nikation mit einem anderen Knoten, der die ANSI-Schlüsselverteilung unterstützt, einrichten. Die CA erlaubt einen Mischkanal (CV oder CV=0), so daß ein einziges KEK-Paar benutzt werden kann und trotzdem aktuelle Anwendungen, die im Kompatibilitätsmodus laufen, und neu entwickelte Anwendungen, die gleichzeitig auf einem CA-Knoten laufen, mittels eines einzigen KEK-Paares mit einem anderen CA-Knoten kommunizieren können.

#### Schlüsselarten und Kanalarten

Unter der CA sind jedem CV eine Art und eine Unterart als codierte Felder im Steuervektor zugeordnet. Der Schlüsselverteilungsmodus, durch den ein Schlüssel legitim übertragen werden kann, wird auch implizit durch die Art/Unterart definiert. (Diese implizite Definition wird anstelle der Codierung der zulässigen Kanalarten in einem separaten Feld des Steuervektors verwendet. Dadurch werden redundante Felder im Steuervektor vermieden.) Der Tabelle in Fig. 70 ist zu entnehmen, welche Kanalarten (d.h. Schlüsselverteilungsmodi) zur Kommunikation mit CV-Arten/-Unterarten verwendet werden können.

#### Deklarierte Schlüsselverteilungsmodus (mittels der Chiffrieranweisung)

In jeder Anweisung, bei der ein Export oder Import von Schlüsseln beteiligt ist (GKS, RFMK und RTMK) muß die Kanalart (oder der Schlüsselverteilungsmodus) durch einen Parameter der Anweisung (Schlüsselverteilungsmodus oder kurz Modus) deklariert werden. Da die ANSI-Chiffrieranweisungen von den anderen CA-Anweisungen getrennt sind, braucht der Modusparameter nur zur Unterscheidung zwischen CV und CV = 0 angegeben zu werden. Der Modusparameter hat als die Form Modus = CV oder Modus = (CV = 0).

#### Verbindungssteuerungsfeld

Bei der CV-Art "KEK" ist ein CV-Feld mit der Bezeichnung "Verbindungssteuerung" definiert. Im Verbindungssteuerungsfeld werden die zulässigen Kanalarten definiert, unter denen der KEK verwendet werden kann. Das Verbindungssteuerungsfeld ist folgendermaßen definiert:

Codierung der Verbindungssteuerung	Interpretation
00	entfällt
01	nur CV
10	nur CV = 0
11	CV oder CV = 0

Das Feld "entfällt" ist für mögliche derzeitige und zukünftige Definitionen von KEK-Unterarten definiert, für die das Verbindungssteuerungsfeld nicht von Bedeutung ist. Für CV-Art/Unterart "KEK/ANSI" beispielsweise hat die Verbindungssteuerung keine Bedeutung, und das Feld wird deshalb auf '00' gesetzt. Verbindungssteuerung = '11' bedeutet, daß mit dem betreffenden KEK Schlüssel entweder über einen CV-Kanal oder über einen CV=0-Kanal übertragen werden können, je nachdem, welcher Verteilungsmodus in der Chiffrieranweisung angegeben wird.

#### Schlüsselverteilungsmodus und Verbindungssteuerung

Der in der Chiffrieranweisung angegebene Schlüsselverteilungsmodus muß immer ein zulässiger Modus sein, das heißt, das Verbindungssteuerungsfeld im Steuervektor des KEK, der zum Chiffrieren des übertragenen Schlüssels benutzt wird, muß den angegebenen Schlüsselverteilungsmodus erlauben. Die Regeln hierfür sind in der Tabelle in Fig. 71 aufgeführt.

#### Allgemeine Prüfregeln

Im bisherigen Teil der Beschreibung wurde ein Regelsatz dargestellt, der nun zusammengefaßt und exemplarisch beschrieben werden kann.

In jeder Chiffrieranweisung, die zum Import oder Export eines Schlüssels führt, sind drei Parameter von Interesse:

- der in der Chiffrieranweisung angegebene Verteilungsmodus (der im folgenden als M bezeichnet werden soll)
- das Verbindungssteuerungsfeld im Steuervektor für den KEK, der zur Schlüsselübertragung verwendet werden soll (im folgenden als L bezeichnet)
- das CV-Art/Unterart-Feld im Steuervektor für den übertragenen Schlüssel (im folgenden als T bezeichnet).

Um zu entscheiden, ob die Anweisung ausgeführt werden kann, werden zwei Prüfungen durchgeführt. (Es werden noch andere Prüfungen durchgeführt, die hier aber nicht relevant sind.) Dabei wird folgendes geprüft:

- T muß durch M erlaubt sein
- M muß durch L erlaubt sein

☐ Das bedeutet, daß für die CV-Art/Unterart die Übertragung über den durch M angegebenen Schlüsselverteilungsmodus erlaubt sein muß, und daß der durch M angegebene Modus von L unterstützt oder zugelassen sein muß.

CV-Art/Unterart	Kanalart	
	CV	CV = 0      ANSI

Daten/Vertraulichkeit	j	n	n
Daten/MAC	j	n	n
Daten/XLT	j	n	n
Daten/Kompatibilität	j*	j	n
Daten/ANSI	n	n	j
KEK/Sender	j	n	n
KEK/Empfänger	j	n	n
KEK/ANSI	n	n	j
PIN/Chiffrieren	j	n	n
PIN/Generieren	j	n	n
ICV/Zwischen-MAC ICV	n	n	n
Schlüsselteil/entfällt	j	n	n

HINWEIS: Ein Schlüssel der Art/Unterart = Daten/Kompatibilität (d.h. ein Kompatibilitäts-Datenschlüssel) kann über einen CV-Kanal mit einem Steuervektor oder über einen CV=0-Kanal durch Entfernen des Steuervektors übertragen werden.

#### Schlüsselverteilungszentrum

Das Schlüsselverteilungszentrum (KDC) ist das Zentrum, in dem Schlüssel generiert und an andere Knoten des Netzwerks verteilt werden. Das KDC kann jedoch kein lokales Exemplar dieser Schlüssel für eine eigene Verwendung behalten.

- Schlüssel an zwei CV-Knoten verteilen

C sein das KDC-Zentrum, das Schlüssel für die zwei Knoten A und B verteilt.  $KK_{Ca}$  sei der Mehrdomänenschlüssel, der verwendet wird, um Schlüssel von C an A zu senden, und  $KK_{Cb}$  sei der Mehr-

domänenschlüssel, der verwendet wird, um Schlüssel von C an B zu senden. In der Anweisung GKS kann im KDC der Export-Export-Modus verwendet werden, um zwei Exemplare eines Schlüssels K zu generieren und an die Knoten A und B zu senden. Die beiden Exemplare des generierten Schlüssels werden unter der Form  $e*KKca.C3(K)$  bzw.  $e*KKcb.C4(K)$  chiffriert und übermittelt, wobei C3 und C4 die Steuervektoren sind, die den Verwendungszweck des Schlüssels K an den Knoten A und B festlegen. Die Steuervektoren C3 und C4 werden vom CFAP anhand der Benutzerangaben generiert. Für die Attribute von C3 und C4 gelten folgende Regeln:

- 1. Wenn der zu verteilende Schlüssel K ein Datenschlüssel ist, müssen folgende Bedingungen erfüllt werden:
  - Als CV-Art-Attribut muß bei beiden Steuervektoren die Hauptart = "Datenschlüssel" angegeben sein, und die Unterart muß bei beiden Steuervektoren identisch sein.
  - C3 und C4 können gleiche oder entgegengesetzte Verwendungsattribute haben. Die beiden Knoten A und B können zum Beispiel den Schlüssel K sowohl zum Chiffrieren als auch zum Dechiffrieren verwenden (gleiche Verwendungsattribute in C3 und C4), oder Knoten A kann den Schlüssel nur zum Generieren eines MAC benutzen, während B ihn nur zum Überprüfen eines MAC benutzen kann (unterschiedliche Verwendungsattribute in C3 und C4). In Fig. 34 sind alle zulässigen Kombinationen der Verwendungsattribute von C3 und C4 aufgeführt.
- 2. Wenn K ein Schlüsselchiffrierschlüssel ist, gilt:
  - Wenn C3 das CV-Art-Attribut "KEK/Sender" hat, muß C4 das CV-Art-Attribut "KEK/Empfänger" haben und umge-

kehrt. Das bedeutet, wenn der Schlüssel K an Knoten A zum Senden von Schlüsseln benutzt wird, muß er an Knoten B zum Empfangen von Schlüsseln verwendet werden und umgekehrt.

--- Das Verbindungssteuerungsattribut muß in beiden Steuervektoren identisch sein.

-- 3. Wenn K ein PIN-Chiffrierschlüssel ist, müssen sowohl C3 als auch C4 das CV-Art-Attribut "PIN/Chiffrieren" aufweisen.

Neben den genannten Attributen setzt das CFAP auch das Exportkontrollattribut in C3 und C4, wobei dieser Wert davon abhängt, was das KDC für den weiteren Export des Schlüssels K durch die Empfangsknoten A und B vorsieht. Die Bedeutung des Exportkontrollfeldes wird im Abschnitt über die Steuervektoren beschrieben.

An den Empfangsknoten A und B kann der Schlüssel K mittels der Anweisung RTMK abgerufen werden. A kann beispielsweise mit RTMK den Schlüssel e\*KKca.C3(K) unter seinen eigenen Hauptschlüssel (nach EXKLUSIV-ODER-Verknüpfung mit einem Steuervektor) umchiffrieren, damit er in diesem System benutzt werden kann.

HINWEIS: Schlüssel, die mittels der Anweisung GKS von einem CV-Knoten an einen anderen CV-Knoten übermittelt werden sollen, müssen über den CV-Kanal gesendet werden. Dies impliziert, daß im Verbindungssteuerungsfeld der Steuervektoren, die den Mehrdomänenschlüsseln KKca und KKcb zugeordnet sind, das Attribut "nur CV" oder "CV oder CV = 0" stehen muß.

- Schlüssel an mehr als zwei CV-Knoten verteilen



Der Hauptzweck der Schlüsselverteilungszentren besteht darin, Schlüssel für die Kommunikation zweier Knoten zu erzeugen. Die Verteilung von Schlüsseln an mehr als zwei Knoten ist nur selten erforderlich. Das KDC ist jedoch in der Lage, Schlüssel an mehr als zwei Knoten zu verteilen, vorausgesetzt, die den zu versendenden Schlüsselexemplaren zugeordneten Steuervektoren sind identisch.

KKci sei der Mehrdomänenschlüssel zum Versenden von Schlüsseln vom Knoten C, der als KDC fungiert, an den Knoten i ( $i = 1, 2, \dots, n$ ). Das KDC kann einen Schlüssel wie folgt an n Knoten ( $n \geq 2$ ) verteilen:

- 1. Mit der Anweisung KGEN wird ein Schlüssel K generiert. K wird unter dem Hauptschlüssel KM des KDC chiffriert. K hat also die Form  $e*KM.C1(K)$ , wobei C1 der Steuervektor ist, der die Attribute von K festlegt. Im Exportkontrollfeld von C1 muß das letzte Bit (Bit 1) den Wert "1" haben, damit K mittels RFMK exportiert werden kann.
- 2. Die Funktion RFMK wird n mal ausgeführt, um  $e*KM.C1(K)$  in die Form  $e*KKci.C2(K)$  (mit  $i = 1, 2, \dots, n$ ) umzuwandeln, wobei C2 der neue Steuervektor ist, der das Attribut des Schlüssels K für die Verwendung am Empfangsknoten angibt. Diese Form wird dann an alle Knoten i gesendet.

An den Empfangsknoten kann der Schlüssel K mit der Anweisung RTMK abgerufen werden.

HINWEIS: Bei dieser Anwendung steht im KDC ein lokales Exemplar des generierten Schlüssels K zur Verfügung. Dies darf nur dann zulässig sein, wenn der Schlüssel in der Anwendung nicht offen-

gelegt wird. Diese Anwendung ist daher nur sehr beschränkt möglich.

Eine Alternative, bei der das KDC keine lokalen Exemplare des generierten Schlüssels benötigt, kann wie folgt implementiert werden:

- 1. Das KDC verwendet die Anweisung GKS im Export-Export-Modus, um Schlüssel an einen oder zwei Knoten zu senden.
  - 2. Das KDC reimportiert ein Exemplar des Schlüssels unter einem KEK mit den Attributen "XLTKEY in" und "XLTKEY out" (vorausgesetzt, die CA erlaubt dies) und wandelt dieses Exemplar in die Formen um, in denen es an die anderen Knoten gesendet werden kann.
- Schlüssel an Nicht-CV-Knoten verteilen

Das KDC kann nur Datenschlüssel an zwei oder mehr Nicht-CV-Knoten verteilen. Die Verteilung anderer Schlüsselarten an Nicht-CV-Knoten ist nicht zulässig. Datenschlüssel werden zuerst mit der Anweisung KGEN generiert und anschließend mit der Anweisung RFMK über den Kompatibilitätskanal an Nicht-CV-Knoten gesendet. Dies bedeutet implizit, daß der Steuervektor, welcher dem KEK zugeordnet ist, unter dem die Schlüssel übertragen werden, im Verbindungssteuerungsfeld das Attribut "CV = 0" enthalten muß. Natürlich muß im Exportkontrollfeld des dem generierten Datenschlüssel zugeordneten Steuervektors der Export des Schlüssels (mittels RFMK) erlaubt sein, damit der Schlüssel verteilt werden kann.

HINWEIS: Wie im obigen Fall ist im KDC ein lokales Exemplar des generierten Schlüssels K verfügbar. Dies darf nur dann zulässig sein, wenn der Schlüssel in der Anwendung nicht offengelegt

wird. Diese Anwendung ist daher nur sehr beschränkt möglich. Die im obigen Fall vorgeschlagene Alternative, GKS und XLTKEY zu kombinieren, ist auch in diesem Fall anwendbar.

- Schlüssel an einen CV-Knoten und einen Nicht-CV-Knoten verteilen

Das KDC kann nur Datenschlüssel sowohl an CV-Knoten als auch an Nicht-CV-Knoten senden. Dies ist dadurch bedingt, daß nur Datenschlüssel an einen Nicht-CV-Knoten übertragen werden können. Die Verteilung kann mittels KGEN und RFMK oder mittels GKS und RFMK erfolgen. Dies bedeutet, daß der Datenschlüssel zuerst mit der Anweisung KGEN oder GKS generiert wird. Ein Exemplar wird dann mittels RFMK oder durch die Anweisung GKS (im OP-EX-Modus) selber an den CV-Knoten gesendet (der CV-Kanal sollte immer benutzt werden). Das andere Exemplar wird mittels RFMK unter CV = 0 an den Nicht-CV-Knoten gesendet.

Auch in diesem Fall ist im KDC ein lokales Exemplar des generierten Schlüssels verfügbar. Mit der Kombination GKS/XLTKEY kann dieses Problem umgangen werden.

#### Schlüsselumwandlung

†

Schlüssel können mit der Funktion "Schlüssel umwandeln" von einem Schlüsselchiffrierschlüssel unter einen anderen Schlüsselchiffrierschlüssel umchiffriert werden. Diese Funktion ist bei Schlüsselumwandlungszentren (KTC) nützlich, d.h. in einer Umgebung, in der ein oder mehrere Knoten zur Umwandlung von Schlüssel für andere Knoten eingesetzt werden. Beispiel: Ein System enthält drei Chiffrierknoten A, B und C wie in Fig. 72. Neben der normalen Funktion als Chiffrierknoten wird C als KTC für A und B eingesetzt. A kann über die Mehrdomänenschlüssel KEKac und KEKca mit C kommunizieren. Entsprechend kommuniziert B über KEKbc und KEKcb mit C. Angenommen, A sendet B Daten, die unter

dem Datenschlüssel KD chiffriert sind, und A und B haben anfangs keine gemeinsamen Mehrdomänenschlüssel für die Kommunikation. Um die Daten zu dechiffrieren, muß B den Schlüssel KD kennen. Da A und B keine gemeinsamen Mehrdomänenschlüssel besitzen, kann A keine chiffrierten Schlüssel senden, die von B rekonstruiert werden können. C kann B helfen KD zu empfangen, indem C als KTC fungiert. Zuerst sendet A ein unter KEKac chiffriertes Exemplar von KD (d.h.  $e*KEKac.C1(KD)$ ) an C. Dann ruft C die Funktion XLTKEY auf, um  $e*KEKac.C1(KD)$  in  $e*KEKcb.C1(KD)$  umzuwandeln, und sendet  $e*KEKcb.C1(KD)$  an B. Im Knoten B wird KD mit Hilfe der Funktion RTMK rekonstruiert, und die chiffrierten Daten können unter Verwendung des empfangenen KD mit der Dechiffrierfunktion dechiffriert werden.

Die Anweisung XLTKEY ist so gestaltet, daß das KTC kein Exemplar des KD behalten kann. Aus Sicherheitsgründen erlaubt die CA auch kein Mischen und Abgleichen von Kanälen, unter denen die umzuwandelnden Schlüssel ankommen (Eingangskanal) bzw. abgehen (Ausgangskanal). Wenn der Schlüssel über einen CV-Kanal ankommt, muß der umgewandelte Schlüssel also auch über einen CV-Kanal abgesendet werden; wenn der Eingangskanal ein CV=0-Kanal ist, muß der Ausgangskanal ebenfalls ein CV=0-Kanal sein. Dies bedeutet, daß die Steuervektoren für die dem Eingangskanal zugeordneten KEKs und die dem Ausgangskanal zugeordneten KEKs (im obigen Beispiel also KEKac und KEKcb) im Verbindungssteuerungsfeld Attribute enthalten müssen, die einer der bei der Anweisung XLTKEY beschriebenen gültigen Kombinationen entsprechen.

#### Export von Schlüsseln

Der Export eines Schlüssels ist die Chiffrierung des Schlüssels unter einer für die Versendung an andere Knoten geeigneten Form. Das heißt, der Schlüssel wird unter einem Schlüsselchiffrierschlüssel chiffriert, den beide kommunizierenden Knoten besitzen.

Ein CV-System kann Schlüssel mit anderen Systemen austauschen, unabhängig davon, ob es sich bei diesen um CV-Systeme handelt oder nicht. In der CA sind für den Export von Schlüsseln folgende Regeln definiert:

- Export von Schlüsseln an ein CV-System

Ein Steuervektorsystem kann Datenschlüssel, Schlüsselchiffrierschlüssel, PIN-Schlüssel, Zwischen-ICVs, Schlüsselteile und Tokens an andere CV-Knoten exportieren.

Der Export an CV-Knoten erfolgt mit der Anweisung RFMK oder mit der Anweisung GKS wie oben beschrieben.

- Export von Schlüsseln mit der Anweisung RFMK

Schlüssel, die in einem System lokal benutzt werden, können mit der Funktion RFMK an andere CV-Knoten exportiert werden, wenn die Angabe im Exportkontrollfeld des betreffenden Steuervektors dies zuläßt. In der Regel werden solche Schlüssel entweder durch die Anweisungen KGEN oder GKS generiert oder von einem anderen Knoten empfangen.

Angenommen, K sei ein lokal im System benutzter Schlüssel, der in der Form  $e \cdot KM.C(K)$  chiffriert ist, wobei C der vom CFAP erstellte und übergebene Steuervektor ist, der die Verwendung von A bestimmt. Unabhängig davon, ob K im System generiert oder von einem anderen Knoten empfangen wurde oder durch eine lokale Umwandlung eines vorhandenen Schlüssels (durch die Anweisung LCVA) entstanden ist, muß die Entscheidung über den Export von K im Exportkontrollfeld des Steuervektors C angegeben sein. Ist im Exportkontrollfeld von C der Export von K an einen anderen CV-Knoten erlaubt, kann K mit der Anweisung RFMK exportiert werden. Angenommen, KK sei der Mehrdomänenschlüssel zur Übertragung von Schlüsseln von A an diesen Knoten. Die Form des zu übertragenden

Schlüssels ist von der für die Übertragung verwendeten Kanalart abhängig. Die Kanalart wird beim Funktionsaufruf im Verteilungsmodus der Anweisung RFMK angegeben. Die Anweisung RFMK wird nur dann ausgeführt, wenn der Verteilungsmodus-Parameter von RFMK und der Wert im Exportkontrollfeld des Steuervektors Ckk (dem Schlüssel KK zugeordneter Steuervektor) eine der in Fig. 71 aufgeführten gültigen Kombinationen bilden.

- Wenn der Verteilungsmodus den CV-Kanal verlangt und im Verbindungssteuerungsfeld von Ckk entweder "CV" oder "CV oder CV = 0" steht, ist die Kombination gültig. K wird in der Form  $e*KK.C1(K)$  exportiert, wobei C1 der Steuervektor ist, der vom CFAP anhand von C erstellt und an die Anweisung RFMK übergeben wurde.
- Wenn der Verteilungsmodus den CV=0-Kanal verlangt und im Verbindungssteuerungsfeld von Ckk entweder "CV = 0" oder "CV oder CV = 0" steht, ist die Kombination gültig. K wird unter der Form  $e*KK(K)$  exportiert. Der Steuervektor wird in diesem Fall abgetrennt. Es ist zu beachten, daß nur Schlüssel der CV-Art "Daten/Kompatibilität" über den CV=0-Kanal übertragen werden dürfen. Andere Schlüssel sind aus Sicherheitsgründen ausgeschlossen (siehe den Abschnitt über Angriffe auf die Sicherheit). Vorhandene Anwendungen (z.B. Anwendungen, die auf IBM CUSP/3848 laufen), die bei der Übernahme auf CA-Knoten keine Steuervektorstrukturen besitzen, verwenden zum Exportieren von Schlüsseln den CV=0-Kanal.

#### - Export von Schlüsseln mittels GKS

Mit der Anweisung GKS im OP-EX-Modus, im OP-IM-Modus, im IM-EX-Modus und im EX-EX-Modus können Schlüssel gleich bei der Generierung an andere CV-Knoten exportiert werden. Die Modi dieser Anweisung werden unter "Elektronische Schlüsselgenerierung" be-

schrieben. In diesen Modi wird nur das Exportexemplar an andere Knoten gesendet; das Importexemplar wird nicht versendet, sondern zu einem späteren Zeitpunkt vom generierenden Knoten selber empfangen, und das lokale Exemplar kann nur im generierenden Knoten lokal benutzt werden.

Im Gegensatz zu der Anweisung RFMK exportiert die Anweisung GKS das Exportexemplar nur unter dem CV-Kanal an einen anderen CV-Knoten. Dies bedeutet, daß das Verbindungssteuerungsfeld des Steuervektors, der dem zum Chiffrieren des Exportexemplars verwendeten Schlüsselchiffrierschlüssel zugeordnet ist, das Attribut "CV" oder "CV oder CV = 0" enthalten muß. Bei der gleichen Notation wie oben wird das Exportexemplar unter der Form `e*KK.C1(K)` exportiert.

Die Funktion GKS bietet eine sichere Möglichkeit, Schlüssel an CV-Systeme zu exportieren. Sie kann nicht direkt zur Versendung von Schlüsseln an Nicht-CV-Systeme benutzt werden.

HINWEIS: Wenn mit RFMK oder GKS ein Schlüssel unter dem CV-Kanal exportiert wird, kann es Fälle geben, in denen der dem Schlüssel zugeordnete Steuervektor nicht mit übertragen wird. Um den richtigen Schlüssel wiederherzustellen, muß der Empfangsknoten in der Lage sein, den Steuervektor korrekt zu rekonstruieren. Unter "Steuervektoren bei der Übertragung" werden für den Fall, daß der Steuervektor nicht mitgesendet wird, Prozeduren zur Erstellung des Steuervektors am sendenden Knoten und zur Rekonstruktion des Steuervektors am empfangenden Knoten beschrieben.

Beim Senden eines Schlüssels kann der sendende Knoten festlegen, ob der empfangende Knoten den Schlüssel an andere Knoten weiterversenden darf oder nicht.

- Wenn der sendende Knoten nicht wünscht, daß empfangende Knoten den Schlüssel an andere Knoten weitergeben darf, setzt

das CFAP den Wert im Exportkontrollfeld des dem exportierten Schlüssel zugeordneten Steuervektors auf B'10'.

- Wenn der sendende Knoten dem empfangenden Knoten die weitere Exportkontrolle dieses Schlüssels erlaubt, setzt das CFAP im Exportkontrollfeld Bit 0 auf '0'; das andere Bit ist nicht relevant. Wenn der Schlüssel am Zielknoten empfangen wird, kann dieser die zwei Bits des Exportkontrollfeldes beliebig setzen.

Die CA erlaubt die Reduzierung der Exportkontrollberechtigung mittels der Anweisung LCVA. Beispiel: Nachdem Knoten A einen Schlüssel K an einen anderen Knoten exportiert hat, soll kein weiterer Export erlaubt sein. Knoten A kann den Export dieses Schlüssels untersagen, indem auf den Schlüssel K und den zugeordneten Steuervektor die Anweisung LCVA angewendet wird, wodurch die Exportkontrollberechtigung auf B'00 oder B'10' (kein Export) herabgesetzt wird.

- Export von Schlüsseln an Nicht-CV-Systeme

Ein direkter Export von Schlüsseln an Nicht-CV-Systeme ist nur mit der Anweisung RFMK möglich, vorausgesetzt, die Angabe im Exportkontrollfeld des zugehörigen Steuervektors erlaubt dies. Aus Sicherheitsgründen können auch nur Schlüssel der CV-Art "Daten/Kompatibilität" an Nicht-CV-Knoten exportiert werden. Wenn mit RFMK ein Schlüssel an einen Nicht-CV-Knoten exportiert wird, kann für die Versendung nur der CV=0-Kanal benutzt werden. Dies bedeutet, daß der Verteilungsmodus-Parameter in der Anweisung RFMK den CV=0-Kanal vorschreiben muß, und daß im Verbindungssteuerungsfeld des Steuervektors, der dem zur Versendung des Schlüssels benutzten Schlüsselchiffrierschlüssel zugeordnet ist, das Attribut "CV = 0" oder "CV oder CV = 0" stehen muß. Bei der gleichen Notation wie oben wird der Schlüssel in der Form e\*KK(K) an einen Nicht-CV-Knoten gesendet.



### Import von Schlüsseln

Der Import eines Schlüssels entspricht der Umchiffrierung des Schlüssels von einem Schlüsselchiffrierschlüssel unter den Hauptschlüssel des Empfangsknotens. Schlüssel in importierbarer Form (d.h. unter einem Schlüsselchiffrierschlüssel chiffrierte Schlüssel) werden entweder von anderen Knoten gesendet oder mit der Anweisung GKS (im OP-IM-Modus oder im IM-EX-Modus) generiert. Zum Importieren eines Schlüssels kann nur die Anweisung RTMK benutzt werden.

#### - Import von CV-Systemen

Ein CV-System kann einen von einem CV-Knoten gesendeten Schlüssel jeder beliebigen CA-Schlüsselart empfangen. Der von einem anderen CV-Knoten gesendete Schlüssel kann über einen CV-Kanal oder über einen CV=0-Kanal ankommen und muß vom entsprechenden Kanal empfangen werden. Die Kanalart wird beim Funktionsaufruf durch den Verteilungsmodus-Parameter in der Anweisung RTMK festgelegt.

-- Wenn der Verteilungsmodus den CV-Kanal vorschreibt, wird der Schlüssel über den CV-Kanal gesendet und empfangen. Der Schlüssel kommt vom sendenden Knoten in der Form  $e*KK.C3(K)$ , wobei KK der gemeinsame Mehrdomänenschlüssel der kommunizierenden Knoten und C3 der dem versendeten Schlüssel K zugeordnete Steuervektor ist. Der Steuervektor C3 kann zusammen mit dem Schlüssel übertragen werden, muß aber nicht.

-- Wenn C3 mit dem Schlüssel zusammen gesendet wird, erstellt das CFAP (am Empfangsknoten) anhand des übertragenen Steuervektors C3 einen neuen Steuervektor C2. Das CFAP übergibt C2, C3 und andere Parameter an die Anweisung RTMK. RTMK rekonstruiert den Schlüssel K mit Hilfe

von C3, prüft dann C2, und chiffriert mit Hilfe von C2 den rekonstruierten Schlüssel unter den Hauptschlüssel um. Der abgerufene Schlüssel liegt jetzt in der für die lokale Verwendung geeigneten Form  $e*KM.C2(K)$  vor.

- Wenn C3 nicht mit dem Schlüssel zusammen gesendet wird, rekonstruiert das CFAP (am Empfangsknoten) zuerst C3, erstellt dann anhand von C3 einen neuen Steuervektor C2 und übergibt diese Steuervektoren und andere Parameter an die Anweisung RTMK. Wie oben wird der Schlüssel in der Form  $e*KM.C23(K)$  abgerufen.

Das CFAP kann, abhängig vom Inhalt des Exportkontrollfeldes von C3, im Exportkontrollfeld von C2 einen anderen Wert setzen als im Exportkontrollfeld von C3.

- Wenn im Exportkontrollfeld von C3 der Wert  $B'1Y'$  steht, wobei Y entweder 0 oder 1 ist, muß das CFAP im Exportkontrollfeld von C2 den gleichen Wert setzen wie im Exportkontrollfeld von C3. Wenn  $Y = 0$  ist, erlaubt der sendende Knoten den weiteren Export des Schlüssels durch den empfangenden Knoten nicht. Ist  $Y = 1$ , kann der Schlüssel an andere Knoten weiterexportiert werden.

- Wenn im Exportkontrollfeld von C3 der Wert  $B'0Y'$  steht, wobei Y entweder 0 oder 1 ist, kann das CFAP im Exportkontrollfeld von C2 einen beliebigen Wert angeben. In diesem Fall können die Exportkontrollfelder von C2 und C3 also unterschiedliche Werte enthalten. Es ist zu beachten, daß das zweite Bit des Exportkontrollfeldes von C3 keinen Einfluß auf den weiteren Export des empfangenen Schlüssels hat. Beispiel: Wenn im Exportkontrollfeld von C3  $B'0Y' = B'00'$  ist (d.h., wenn  $Y = 0$  ist), kann der Empfangsknoten den weiteren Export des empfangenen

Schlüssels erlauben, indem im Exportkontrollfeld von C2 entweder B'01' oder B'11' gesetzt wird.

HINWEIS: Die Funktion RTMK wird nur ausgeführt, wenn im Verbindungssteuerungsfeld des KK zugeordneten Steuervektors entweder das Attribut "CV" oder das Attribut "CV oder CV = 0" steht.

- Wenn der Verteilungsmodus den CV=0-Kanal vorschreibt, wird der Schlüssel über den CV=0-Kanal gesendet und empfangen. Der Schlüssel kommt vom sendenden Knoten in der Form e\*KK(K) an. Am Empfangsknoten wird der Schlüssel in der Form e\*KM.C2(K) abgerufen, wobei C2 der K zugeordnete Steuervektor ist, der vom CFAP erstellt und an RTMK übergeben wird. Im Exportkontrollfeld von C2 kann vom CFAP jeder gewünschte Wert gesetzt werden.

HINWEIS: Nur die Schlüsselart "Daten/Kompatibilität" wird über den CV=0-Kanal gesendet und empfangen. C2 hat also die CV-Art "Daten/Kompatibilität", und die Verwendungsattribute sind auf die Kombinationen von "Chiffrieren", "Dechiffrieren", "MAC generieren" und "MAC überprüfen" beschränkt. Es ist außerdem zu beachten, daß die Anweisung RTMK nur ausgeführt werden kann, wenn im Verbindungssteuerungsfeld des KK zugeordneten Steuervektors das Attribut "CV = 0" oder "CV oder CV = 0" angegeben ist.

#### Empfangen von Schlüsseln von Nicht-CV-Systemen

Ein CV-System kann von einem Nicht-CV-System nur die Schlüsselart "Daten/Kompatibilität" empfangen. Der gesendete Schlüssel kommt vom sendenden Knoten nur unter einem CV=0-Kanal an (in der Form e\*KK(K)). Der Schlüssel wird auf die gleiche Weise empfangen wie wenn er über einen CV=0-Kanal von einem CV-Knoten gesendet wird.

### Steuervektoren bei der Übertragung

#### Einführung in die Übertragungsprotokolle für die Übermittlung von Schlüsseln

Die CA enthält Übertragungsprotokolle für die elektronische Übermittlung von Chiffrierschlüsseln von einer Chiffriervorrichtung an eine andere. Das CA-Übertragungsprotokoll enthält eine Steuervektordefinition für die Übertragung und eine Angabe der zulässigen Protokolle zur Verwendung dieser Steuervektordefinition für die elektronische Übermittlung von Chiffrierschlüsseln.

Grundsätzlich liefert das CA-Übertragungsprotokoll ein Mittel zur kryptographischen Trennung der elektronisch übermittelten Schlüssel nach Schlüsselart oder Verwendungszweck. Das Übertragungsprotokoll ist so gestaltet, daß wenn bei der Übertragung eine Schlüsselart durch eine andere ersetzt wird, der ersetzende Schlüssel beim Empfänger nicht korrekt rekonstruiert werden kann. In diesem Fall haben die sendende und die empfangende Chiffriervorrichtung keine zusammenpassenden Schlüssel, so daß eine Kommunikation mit einem solchen Schlüsselpaar nicht möglich ist. Ein Gegner kann bestenfalls das Chiffriersystem stören. Angriffe, bei denen ein Gegner versucht, eine empfangende Chiffriervorrichtung zur Benutzung eines Schlüssels in einer nicht von der sendenden Chiffriervorrichtung vorgesehenen Weise zu zwingen, werden abgewehrt.

#### Anforderungen an die Schlüsselverwaltung der kommunizierenden Chiffriervorrichtungen

Das CA-Übertragungsprotokoll stellt folgende Anforderungen an die Schlüsselverwaltungen zweier miteinander kommunizierender Chiffriervorrichtungen i und j:

- 1. i und j müssen ein Schlüsselchiffrierschlüsselpaar KEKij und KEKji teilen, wobei KEKij zum Senden von Schlüsseln von i an j und KEKji zum Senden von Schlüsseln von j an i benutzt wird und KEKij und KEKji doppelte Länge (128 Bit) aufweisen. Wahlweise können i und j nur KEKij zum Senden von Schlüsseln von i an j oder KEKji zum Senden von Schlüsseln von j an i besitzen. Außerdem können i und j wahlweise zusätzliche Schlüsselchiffrierschlüssel oder Schlüsselpaare für Zwecke der elektronischen Schlüsselverteilung gemeinsam haben. Diese Schlüsselchiffrierschlüssel werden gewöhnlich als Mehrdomänenschlüssel bezeichnet.
- 2. An i und j müssen die Schlüssel KEKij und KEKji so "markiert" sein, daß die Verwendung dieser Schlüssel für den elektronischen Export und Import von Schlüsseln (mittels GKS, RFMK und RTMK) nur in Verbindung mit einem Steuervektor möglich in der von der Architektur definierten Weise möglich ist. Zusätzliche Schlüsselchiffrierschlüssel oder Schlüsselpaare, die die Chiffriereinheiten wahlweise gemeinsam haben können, müssen diese Voraussetzung ebenfalls erfüllen.

**HINWEIS:** i und j können einen oder mehrere Schlüsselchiffrierschlüssel (oder Schlüsselpaare) für die elektronische Schlüsselverteilung gemeinsam haben, wobei die gemeinsamen Schlüssel ein Verfahren benutzen, bei dem keine Steuervektoren beteiligt sind. Dies kann notwendig sein, um die Kompatibilität mit vorhandenen Hardware- oder Softwareeinrichtungen an i und/oder j aufrechtzuerhalten. Diese Schlüssel müssen jedoch so "markiert" sein, daß bei der Verwendung dieser Schlüssel zum elektronischen Export und Import von Schlüsseln (mittels GKS, RFMK und RTMK) die Steuervektoren abgetrennt werden wie von der CA vorgeschrieben.

HINWEIS: Werden Steuervektoren in der Chiffriervorrichtung intern und wie durch den gemeinsamen Chiffrierfunktionssatz definiert benutzt, ist bereits ein Verfahren zur "Markierung" der KEKs (wie unter Punkt "b" gefordert) festgelegt. Andernfalls muß die "Markierung" auf andere Weise erfolgen.

Format für die elektronische Übermittlung von Schlüsseln mittels Steuervektoren

Alle Schlüssel, die auf elektronischem Weg von einer Chiffriervorrichtung, einer Schlüsselgenerierungsvorrichtung, einer Schlüsselverteilungsvorrichtung oder einer Schlüsselumwandlungsvorrichtung an eine empfangende Chiffriervorrichtung gesendet werden, müssen mittels des im folgenden beschriebenen Formats chiffriert werden:

Angenommen,

\*KEK = 128-Bit-Schlüssel, den die sendende und die empfangende Vorrichtung zum Zweck der Übermittlung von Schlüsseln von der sendenden an die empfangende Vorrichtung teilen  
C = 64-Bit-Steuervektor  
K = zu übertragender 64-Bit-Schlüssel  
K1, K2 = zu übertragender 128-Bit-Schlüssel

Dann gilt für Schlüssel einfacher und doppelter Länge folgendes Format:

Zu übertragender Schlüssel	Schlüsselformat
K	e*KEK.C(K)K
K1, K2	e*KEK.C(K1), e*KEK.C2(K2)

Protokoll für Kompatibilität

Das CA-Übertragungsprotokoll definiert zur Wahrung der Kompatibilität mit vorhandenen IBM Verschlüsselungsprodukten wie 3848/CUSP, PCF und 4700 auch einen Standard-Steuervektor, der aus 64 Nullen besteht. In diesem Übertragungsmodus können Schlüssel mit einem Null-Steuervektor (d.h. effektiv ohne Steuervektor) gesendet und empfangen werden. In diesem Fall wird ein Schlüssel K nicht in der Form eKEK.C(K) übertragen, sondern in der Form eKEK(K).

#### Meldungsformate

Das CA-Übertragungsprotokoll erlaubt die Übertragung chiffrierter Schlüssel in zwei Formaten. Es wird davon ausgegangen, daß die gesendete Information in einer Meldung enthalten ist. Das Übertragungsprotokoll enthält gegenwärtig keine Definition für Meldungsformate, da es zahlreiche mögliche Meldungsformate geben könnte, die aufgrund von Anforderungen, die in diesem Dokument nicht vorhergesehen oder definiert werden können, akzeptabel oder wünschenswert sind. Das Übertragungsprotokoll definiert nur die Information, die in einer solchen Meldung gesendet werden muß, je nachdem, welches Format gewählt wird.

#### Erstes Format:

- a. Für 64-Bit-K: C, e\*KEK.C(K)
- b. Für 128-Bit-K1,K2: C, e\*KEK.C(K1), e\*KEK.C(K2)

#### Zweites Format:

- a. Für 64-Bit-K: e\*KEK.C(K)
- b. Für 128-Bit-K1,K2: e\*KEK.C(K1), e\*KEK.C(K2)

HINWEIS: Der Sonderfall CV = 0 wird von den obigen beiden Formaten ebenfalls erfaßt.

### Erstes Format - mit Steuervektor

Das erste Format gilt für den Fall, daß der Steuervektor zusammen mit dem chiffrierten Schlüssel übertragen wird. Da eine empfangende Chiffrieranwendung oder ein empfangendes Verschlüsselungsprogramm immer wissen muß, für welchen Zweck der empfangene Schlüssel bestimmt ist, kann der Steuervektor ein nützliches und kompaktes Mittel sein, alle erforderlichen Informationen über die Verwendung des Schlüssels von einer sendenden an eine empfangende Station zu übermitteln. Darüber hinaus kann ein empfangener Steuervektor anschließend vom empfangenden Anwendungsprogramm an das Verschlüsselungsprogramm (z.B. IBM CUSP) übergeben werden, und von somit abhängig davon, wie die Funktion RTMK implementiert ist, direkt als Parameter der Funktion RTMK vom Verschlüsselungsprogramm an die Chiffriervorrichtung (d.h. die Hardware) übergeben werden. Die Verwaltung des Steuervektors durch die Software (d.h. das CFAP) ist so wahrscheinlich weniger komplex.

### Zweites Format - ohne Steuervektor

Beim zweiten Format sind zwei Fälle möglich. Im ersten Fall haben die sendende und die empfangende Station vorher keine Vereinbarung über die Rekonstruktion des Steuervektors getroffen. In diesem Fall führt die empfangende Station eine Folge vorgeschriebener Schritte zur Rekonstruktion des Steuervektors aus, wobei bestimmte Standardwerte für den Steuervektor benutzt werden. Im zweiten Fall haben die sendende und die empfangende Station vorher eine Vereinbarung über den Steuervektor getroffen. In diesem Fall rekonstruiert die empfangende Station den Steuervektor nach dem zuvor vereinbarten privaten Regelsatz.

Der grundlegende Unterschied zwischen dem ersten und dem zweiten Fall des zweiten Formats besteht darin, daß die empfangende Station den Steuervektor im ersten Fall mit Hilfe von vom Herstel-



ler angegebenen Standardannahmen rekonstruiert, und im zweiten Fall mit Hilfe eines privat vereinbarten Regelsatzes.

### Steuervektorspezifikation

Die Steuervektorspezifikation für die Übertragung entspricht der im Abschnitt "Schlüsselverwaltung" beschriebenen internen Steuervektorspezifikation. Dies bedeutet, daß die durch die CFAP-Makrodefinitionen und CF-Anweisungen vorgegebenen Regeln für die Erstellung und Änderung von Steuervektoren ebenfalls in beiden Fällen identisch sind. Diese Regeln sollen die reibungslose Rekonstruktion von Schlüsseln erleichtern. Der Leser wird daran erinnert, daß bei der Schlüsselrekonstruktion durch Dechiffrierung jedes Bit im Steuervektor exakt den gleichen Wert haben muß wie ursprünglich für die Chiffrierung des Schlüssels angegeben.

Dabei handelt es sich im wesentlichen um folgende Regeln:

- Im allgemeinen ist das CFAP für die Erstellung aller Steuervektoren zuständig, und in manchen Fällen auch für die Prüfung/Erzwingung bestimmter Bitwerte im Steuervektor. Die Chiffriervorrichtung ist in der Regel für die Prüfung/Erzwingung der Bitwerte im Steuervektor zuständig.
- Bei der Erstellung eines Steuervektors müssen alle Bits im Steuervektor angegeben werden. Dabei wird folgendermaßen vorgegangen:
  - Alle reservierten Bits werden auf Null gesetzt. (Dies ist leicht zu erreichen, indem anfänglich  $CV = 0$  gesetzt wird.)
  - Die Antivariantenbits oder die feststehenden Bits werden gesetzt.
  - Die übrigen Bits mit Ausnahme der 8 Paritätsbits (d.h. die unter der Architektur definierten Paritätsbits) wer-

den durch die Parameterwerte gesetzt, die von der Anwendung im sogenannten CFAP-Makro oder gemäß den im Makro vorgegebenen Standardwerten angegeben werden.

- Wenn ein Schlüssel in der Form eKEK.C(K) von einem Sender an einen Empfänger übertragen wird, wobei C nicht der Sonderfall CV = 0 ist und nicht mit übertragen wird, sind beim Empfänger folgende Standardwerte für den Steuervektor definiert:

- Standard-CV-Version = 0
- Standard-Schlüsselart = "Datenschlüssel"
- Standard-Exportkontrolle = "0000", d.h. ohne Beschränkung
- Bei Schlüsselart = "Datenschlüssel": Standard-Verwendungsbits = "1100", d.h. E und D
- Bei Schlüsselart = "KEK/Sender": Standard-Zielsystem = "00", d.h. "CV oder CV = 0", Standard-Verwendungsbits = "1111", d.h. ohne Beschränkung

---

- Bei Schlüsselart = "KEK/Empfänger": Standard-Zielsystem = "00", d.h. "CV oder CV = 0", Standard-Verwendungsbits = "11", d.h. ohne Beschränkung
- Bei Schlüsselart = "PIN/Chiffrieren": Standard-Verwendungsbits = "010101", d.h. PIN-Block Schlüssel ein und Schlüssel aus reformatieren und PIN überprüfen.
- Bei Schlüsselart = "PIN/Generieren": Standard-Verwendungsbits = "01", d.h. PIN überprüfen. Diese Standardwerte sollten in den Anweisungen GKS, RFMK, XLTKY, KGEN, EMK und RTMK verwendet werden.

- Von der CA vorgegebene CV-Standardwerte

Bekannt	Unbekannt	Zu verwendende Werte
Nichts	Ob Sender ein CV-System ist	CV = 0

Sender ist CV-System	CV-Versionsnummer	Version = 00
Sender ist CV-System	Reservierte Bits	lauter binäre Nullen
Sender ist CV-System	CV-Art	CV-Art = "Daten"
CV-Art = DATA	Attribute	E D
CV-Art = PINE	Attribute	RPB-I RPB-O VP
CV-Art = PING	Attribute	VP
CV-Art = KEKS	Attribute	GKS-LR GKS-RR RFMK XLT-O
CV-Art = KEKS	Zielsystem	CV oder CV = 0
CV-Art = KEKR	Attribute	RTMK XLT-I
CV-Art = KEKR	Zielsystem	CV oder CV = 0
CV-Art = KEYP	Attribute	(keine Standardwerte)
CV-Art = IICV	(keine Attribute vorhanden)	

### Speicherung der Schlüssel

Alle Schlüssel außerhalb der Chiffriervorrichtung werden in chiffrierter Form verwaltet. Die Chiffrierung erfolgt unter einem Schlüssel, der durch EXKLUSIV-ODER-Verknüpfung eines Steuervektors mit dem Hauptschlüssel gebildet wird.

Außerhalb der Chiffriervorrichtung gespeicherte chiffrierte Schlüssel können wahlweise zusammen mit dem zur Chiffrierung des gespeicherten Schlüssels verwendeten Steuervektor gespeichert werden. Die Speicherung des unchiffrierten Steuervektors ist jedoch nicht erforderlich, solange dieser anhand anderer mit dem Schlüssel gespeicherter Informationen oder aus dem Kontext, in dem der chiffrierte Schlüssel benutzt wird, dynamisch rekonstruiert werden kann.

Alle generierten und importierten Schlüssel werden im Massenschlüsselspeicher außerhalb der Chiffriervorrichtung chiffriert und gespeichert. Der Schlüsselspeicher ist in zwei separate Bereiche unterteilt: KKDS (KEK-Datei) und DKDS (Datenschlüsseldatei). Zwischen diesen beiden Speicherbereichen wird unterschieden, weil in der CA die Datenschlüssel und die Schlüsselchiffrierschlüssel (sowie die PIN-Schlüssel) mit verschiedenen Mechanismen verwaltet werden.

Chiffrierte Schlüsselchiffrierschlüssel, PIN-Chiffrierschlüssel und PIN-Generierungsschlüssel können in einer KKDS gespeichert und über nicht geheime Schlüsselkennsätze adressiert werden.

Chiffrierte Datenschlüssel können in einer DKDS gespeichert werden. Datenschlüssel können auch unchiffriert in der Chiffriervorrichtung gespeichert werden, sofern die auf diese Schlüssel zugreifenden Anwendungen dafür geheime Token-Werte mit ca. 56 unabhängigen geheimen Bits verwenden. Zusätzliche Sicherheit für

die in der DKDS gespeicherten chiffrierten Schlüssel kann erreicht werden, indem jeder chiffrierte Schlüssel durch EXKLUSIV ODER mit einer Maske verknüpft wird, die einer Einwegfunktion des zum Zugriff auf den chiffrierten Datenschlüssel verwendeten Tokens entspricht.

In Fig. 73 ist die KKDS mit einigen KEK-Einträgen dargestellt. Jeder Schlüsselchiffrierschlüssel (hier als KEKi bezeichnet) besteht aus zwei 64 Bit langen Schlüsselhälften und wird in Form von zwei 64-Bit-Schlüsselhälften chiffriert und gespeichert. Die erste Hälfte ist die Chiffrierung der linken Schlüsselhälfte (KEKiL) unter dem durch  $KM \text{ XOR } CL$  gebildeten Schlüssel, wobei CL der KEKiL zugeordnete Steuervektor ist. Entsprechend ist die zweite Hälfte die Chiffrierung der rechten Schlüsselhälfte (KEKiR) unter dem durch  $KM \text{ XOR } CR$  gebildeten Schlüssel, wobei CR der KEKiR zugeordnete Steuervektor ist. Die Steuervektoren CL und CR sind bis auf den Wert im Schlüsselformfeld identisch; dort wird angegeben, ob es sich um die rechte oder um die linke Hälfte handelt.

HINWEIS: Ist KEKi ein kurzer Schlüssel (d.h. ein Schlüssel einfacher Länge), wird er dennoch in Form von zwei 64-Bit-Hälften gespeichert. In diesem Fall sind die beiden Hälften identisch, da  $KEKiL = KEKiR$  und  $CL = CR = C$  ist. Die Kopplung der beiden Schlüsselhälften mit dem entsprechenden Steuervektor, der angibt, um welche Schlüsselhälfte es sich handelt, ist zum Schutz vor Angriffen, die als "Schlüsselhälftenverdopplung" bekannt sind, unbedingt erforderlich. Wenn es keine derartige Kopplung gibt, kann ein Gegner eine chiffrierte Hälfte durch die andere ersetzen (d.h. eine Schlüsselhälfte verdoppeln), so daß die beiden Hälften des chiffrierten Schlüssels identisch sind. Er kann dann Chiffrieroperationen mit der Schlüsselhälfte durchführen, um die Suche nach der Schlüsselhälfte zu erschöpfen. Anschließend geht er mit der anderen Schlüsselhälfte ebenso vor. Im wesentlichen ist der Arbeitsfaktor eines langen Schlüssels nun auf

die Größenordnung des Arbeitsfaktors eines kurzen Schlüssels reduziert.

Bei allen Chiffrieroperationen ruft das CFAP die chiffrierten Schlüssel aus dem Schlüsselspeicher ab und übergibt sie an die auszuführende Anweisung. Diese Schlüssel werden von der Anweisung nur innerhalb der CF dechiffriert und benutzt. Sie erscheinen niemals unchiffriert außerhalb der CF.

HINWEIS: Die meisten Schlüsselverwaltungsanweisungen, die in der Regel mit Schlüsselchiffrierschlüsseln arbeiten, müssen zweimal aufgerufen werden, um beide Hälften eines Schlüsselchiffrierschlüssels doppelter Länge abzuarbeiten bzw. zu erzeugen. Jeder Aufruf verarbeitet oder erzeugt eine Hälfte des Schlüsselchiffrierschlüssels. Das CFAP oder die Anwendung muß deshalb geeignete Parameterwerte (wie die chiffrierte Schlüsselhälfte und den zugeordneten Steuervektor mit dem richtigen Wert im Schlüsselformfeld usw.) angeben, um korrekte Ergebnisse zu erzielen.

In Systemen mit hohem Durchsatz, wo häufig auf Schlüssel zugegriffen wird, können Cache-Mechanismen implementiert werden, um die Zugriffsgeschwindigkeit zu erhöhen.

#### Angriffe von außen oder von innen

Die Schlüsselverwaltung hat das Ziel, Schutz vor Angriffen von außen und vor bestimmten Angriffen von innen zu gewähren.

#### Schlüsselverwaltung gemäß ANSI X9.17

Die Schlüsselverwaltung soll neben der Schlüsselverwaltung gemäß ANSI X9.17 existieren können, ohne die Sicherheit einer der beiden Schlüsselverwaltungen zu beeinträchtigen.

#### Kompatibilität mit vorhandenen Produkten

Die Kompatibilität mit vorhandenen Produkten wie z.B. IBM 3848/CUSP wird auf zwei Ebenen realisiert. Eine Kombination von KGEN, RTMK und RFMK erlaubt die Erzeugung von Datenschlüsseln in vier verschiedenen Formen, die direkt den vier verschiedenen Formen entsprechen, die mit dem derzeitigen GENKEY-Makro des IBM 3848/CUSP erzeugt werden können. Um beispielsweise bei der Generierung eines Sitzungsschlüssels mit dem IBM 3848/CUSP kompatibel zu sein, kann mit der Funktion KGEN eine Zufallszahl generiert werden, die als unter dem Hauptschlüssel chiffrierter Sitzungsschlüssel aufgefaßt wird. Dieser kann dann mit der Funktion RFMK an andere Knoten gesendet werden. Entsprechend kann ein Dateischlüssel auf kompatible Weise generiert werden, indem zuerst mit KGEN eine Zufallszahl erzeugt wird, die als unter einem Dateihauptschlüssel chiffrierter Dateischlüssel aufgefaßt wird, und dieser dann mit der Funktion RFMK in einen unter dem Hauptschlüssel chiffrierten Dateischlüssel zur Verwendung im System umgewandelt wird. Die Funktionen RTMK und RFMK erlauben effektiv einen Spezialfall des Imports und Exports von Datenschlüsseln mit einem Steuervektor aus 64 Nullbits (d.h. ohne Steuervektor).

#### Software-Schnittstelle

Das CFAP bildet die Anwendungsschnittstelle zur Chiffriervorrichtung. Die Schnittstelle kann in Form von Makros oder Unterprogrammaufrufen implementiert sein und eine Schlüsselspeicher-Verwaltungskomponente enthalten, die die Speicherung und den Abruf der in einem Schlüsselspeicher gespeicherten Schlüssel übernimmt.

#### Schlüsselverwaltung gemäß ANSI X9.17

- ANSI-Knotenarchitektur

In Fig. 74 sind die wichtigsten Komponenten eines ANSI X9.17-Knotens dargestellt. Diese Architektur basiert auf den elementaren Systemdiagrammen in Fig. 1, 2 und 3.

Die ANSI-Schlüsselverwaltung ist ein Anwendungsprogramm, das die Chiffrierservicefunktionen des CFAP benutzt, um Schlüssel zu generieren, importieren und exportieren und um die Identifikation von Chiffrierservice-Meldungen (CSMs) zu überprüfen. CSMs sind nach ANSI X9.17 definierte Meldungen zum Austausch von Verschlüsselungsmaterial oder zugehörigen Informationen zwischen kommunizierenden ANSI-Knoten.

Das CFAP verwendet die Servicefunktionen einer Schlüsselspeicherverwaltung zum Zugriff auf ANSI-Schlüssel, die außerhalb der Chiffriervorrichtung (CF) gespeichert sind. Der Speicherbereich ist als Schlüsselspeicher bekannt. Solche Schlüssel werden immer in einer unter dem Systemhauptschlüssel (KM) in Verbindung mit einem zugehörigen Steuervektor chiffrierten Form gespeichert. Die CF stellt dem CFAP die elementaren Chiffrierfunktionen (Chiffrieren, Dechiffrieren, MAC-Generierung usw.) zur Verfügung. Außerdem stellt sie in begrenztem Umfang internen Speicher für den unchiffrierten KM und einige Arbeitsschlüssel zur Verfügung.

Die ANSI-Schlüsselverwaltung nutzt die vom System bereitgestellten Übertragungsservicefunktionen (z.B. VTAM oder Gerätetreiber) zum Senden und Empfangen von CSMs zu bzw. von anderen ANSI-Knoten.

#### - Funktionsschnittstellen von ANSI-Knoten

In Fig. 75 sind einige Funktionsschnittstellen zwischen Komponenten eines ANSI-Knotens dargestellt. Die ANSI-Schlüsselverwaltung greift durch Makroaufrufe auf CFAP-Servicefunktionen zu. Das CFAP benutzt zum Abrufen und Speichern von ANSI-Schlüsseln



Funktionsaufrufe der Schlüsselspeicherverwaltung. Die Schlüsselspeicherverwaltung verwendet seinerseits System-E/A-Servicefunktionen (oder Speicherzugriffsanweisungen) zum Abrufen/Speichern der angeforderten Schlüssel aus dem als Schlüsselspeicher bezeichneten Sekundärspeicher (oder Primärspeicher). Die Schlüsselspeicherverwaltung kann auch für das CFAP die Schlüsselintegrität prüfen. Jeder Schlüsseleintragssatz im Schlüsselspeicher kann aus einem Kennsatz, dem chiffrierten Schlüssel, dem expliziten Steuervektor (CV), unter dem er gespeichert ist, den Send- und Empfangszählern (nur bei KEKs) und anderen schlüsselbezogenen Parametern bestehen. Das CFAP verweist zur Ausführung elementarer Chiffrierfunktionen auf CF-Anweisungen.

- Schlüsselverteilungsumgebungen

ANSI X9.17 unterstützt im wesentlichen drei Schlüsselverteilungsumgebungen: Punkt-zu-Punkt-Umgebungen, Schlüsselverteilungszentren und Schlüsselumwandlungszentren.

-- Punkt-zu-Punkt-Umgebung (P-P)

In dieser Umgebung will ein Knoten (Knoten B) mit einem anderen Knoten (Knoten A) kommunizieren, kann aber keine Datenschlüssel generieren oder abrufen. Es wird jedoch angenommen, daß die Knoten A und B bereits einen manuell installierten Schlüsselchiffrierschlüssel (\*)KKab gemeinsam haben. (Die Notation '(\*)' bedeutet, daß es sich um einen Schlüssel einfacher Länge, KK, oder um einen Schlüssel doppelter Länge, \*KK, handeln kann.) Knoten A generiert auf Anfrage von Knoten B einen oder mehrere Schlüssel, speichert ein lokales Exemplar und gibt die unter (\*)KKab chiffrierten Duplikate an Knoten B weiter.

In Fig. 76 sind zwei ANSI-Knoten in einer Punkt-zu-Punkt-Umgebung dargestellt. Die Knoten besitzen einen gemeinsamen Schlüsselchiffrierschlüssel \*KKab, der zusammen mit den be-

treffenden Sende- und Empfangszählern im Schlüsselspeicher beider Knoten gespeichert ist. Es ist zu beachten, daß  $e*KM.C1(*KKab)$  tatsächlich als  $e*KM.C1(KKabl)$ ,  $e*KM.C1(KKabR)$ , also linke und rechte 64 Bit von  $*KKab$  gespeichert ist. Der im Schlüsselspeicher gespeicherte Datensatz für diesen Schlüssel kann außerdem andere schlüsselbezogene Parameter enthalten.

In diesem Diagramm ist eine Folge von CSM-Übertragungen in einer Punkt-zu-Punkt-Schlüsselverteilungsumgebung dargestellt. Knoten B fordert von Knoten A mittels RSI (Service-Initialisierungsanforderung) einen oder mehrere Schlüssel an. Knoten A sendet den/die angeforderten Schlüssel mittels KSM (Schlüssel-service-Meldung) an Knoten B, und Knoten B quittiert den Empfang mit einer RSM (Antwort-Servicemeldung). Andere CSM-Transaktionen sind in der Punkt-zu-Punkt-Umgebung ähnlich definiert, damit Anforderungen zur Fehlerbehandlung und zur Nicht-Weiterverwendung von Schlüsseln unterstützt werden.

Angenommen, Knoten B fordert von Knoten A einen einzelnen Datenchiffrierschlüssel KD an. Das Meldungsformat einer möglichen RSI für diese Anforderung sieht dann folgendermaßen aus:

```
CSM(MCL/RSI
RCV/KnotenA
ORG/KnotenB
SVR
EDC/aKDX(MCL/RSI ... SVR/))
```

HINWEIS:  $aKDj(X)$  steht für den Meldungs-Identifikationsüberprüfungscode (MAC) von X mittels Schlüssel  $KDj$ . KDX ist ein Sonderfall von  $KDj$ : der feststehende, nicht geheime Hexadezimalschlüssel '0123456789ABCDEF'.

Knoten A generiert den einzelnen KD, chiffriert ihn unter dem mit Knoten B gemeinsamen  $*KKab$ , unterzieht ihn einem Offset mit

dem \*KKab zugeordneten Sendezähler und bildet die an den Knoten B übermittelte KSM:

```
CSM(MCL/KSM
RCV/KnotenB
ORG/KnotenA
KD/e*KKab+Send_Ctr_*KKab(KD)
CTP(Send_Ctr_*KKab
MAC/aKD(MCL/KSM ... CTP/Send_Ctr_*KKab))
```

Hier ist  $e*KKab+Send\_Ctr\_*KKab(KD)$  der Chiffretext, der durch dreifache Chiffrierung von KD unter dem durch Offset von \*KKab mit dem aktuellen Sendezählerwert  $Send\_Ctr\_*KKab$  gebildeten Schlüssel entstanden ist. Die Offset-Funktion wird in Abschnitt 7.4 von ANSI X9.17-1985 beschrieben.

Knoten B empfängt die KSM, extrahiert das KD/-Feld, rekonstruiert den neuen KD und speichert ihn unter dem Hauptschlüssel von Knoten B im Schlüsselspeicher. Anschließend formuliert Knoten B folgende Quittierungs-RSM an Knoten A:

```
CSM(MCL/RSM
RCV/KnotenA
ORG/KnotenB
MAC/aKD(MCL/RSM ... ORG/KnotenB))
```

- Schlüsselverteilungszentrum (KDC)

In Fig. 77 sind drei ANSI-Knoten in einer KDC-Umgebung dargestellt.

In dieser Umgebung will ein Knoten B mit einem Knoten A kommunizieren, mit dem er keine Schlüssel gemeinsam hat. Es wird jedoch angenommen, daß Knoten B einen manuell installierten Schlüsselschiffrierschlüssel (\*)KKbc mit Knoten C gemeinsam hat, der als

Schlüsselverteilungszentrum fungiert. Ferner hat Knoten C einen manuell installierten Schlüsselchiffrierschlüssel (\*)KKac mit Knoten A gemeinsam. Knoten A übergibt die Schlüssel Anforderung von Knoten B an Knoten C, der zwei doppelte Schlüsselsätze generiert. Der erste Satz wird unter (\*)KKac chiffriert, der zweite unter (\*)KKbc. Beide Sätze werden dann an Knoten A zurückgesendet. Knoten A rekonstruiert und speichert den ersten Schlüsselsatz und gibt den zweiten an Knoten B weiter. Knoten B rekonstruiert und speichert seinen Schlüsselsatz genauso.

- Schlüsselumwandlungszentrum (KTC)

In Fig. 78 sind drei ANSI-Knoten in einer KTC-Umgebung dargestellt.

In dieser Umgebung will ein Knoten B mit einem Knoten A kommunizieren, mit dem er keine Schlüssel gemeinsam hat. Es wird jedoch angenommen, daß Knoten A Schlüssel generieren oder abrufen kann und einen manuell installierten Schlüsselchiffrierschlüssel (\*)KKac mit Knoten C gemeinsam hat, der als Schlüsselumwandlungszentrum fungiert. Ferner hat Knoten C einen manuell installierten Schlüsselchiffrierschlüssel (\*)KKbc mit Knoten B gemeinsam. Knoten A generiert den angeforderten Schlüsselsatz, speichert ein lokales Exemplar, und sendet ein unter (\*)KKac chiffriertes Duplikat an Knoten C. Knoten C rekonstruiert den Schlüsselsatz, chiffriert ihn unter (\*)KKbc um und sendet den umgewandelten Schlüsselsatz an Knoten A zurück. Knoten A sendet den Schlüsselsatz an Knoten B, wo er rekonstruiert und gespeichert wird.

- Schlüsselverteilungsszenarien

Im folgenden werden die in ANSI X9.17 definierten Schlüsselverteilungsszenarien kurz umrissen. Die Szenarien werden tabella-

risch dargestellt, und zwar einmal für Schlüsselchiffrierschlüssel und einmal für Datenchiffrierschlüssel.

Jede Tabelle ist in drei Umgebungen untergliedert: Punkt-zu-Punkt-Umgebung, Schlüsselumwandlungszentrum und Schlüsselverteilungszentrum. Für jeden Schlüsselaustausch trifft nur eine Umgebung zu. Ein Szenario für eine bestimmte Umgebung wird im allgemeinen von links nach rechts und von oben nach unten dargestellt. Wo Wahlmöglichkeiten vorhanden sind, wird die betreffende Spalte horizontal unterteilt, so daß die Auswahl und das entsprechende Ergebnis abgelesen werden können. Die CA unterstützt nicht alle ANSI X9.17-Optionen. Nicht unterstützte Optionen sind in der Spalte "SUP?" mit "N" gekennzeichnet.

Die Spalten haben folgende Bedeutung:

- |                      |   |
|----------------------|---|
| -- ENV               | ANSI X9.17-Schlüsselverteilungsumgebung: Punkt-zu-Punkt, Schlüsselverteilungszentrum oder Schlüsselumwandlungszentrum.  |
| -- NOD               | Teilnehmer am Verteilungsszenario. Der Konvention entsprechend ist Knoten B typischerweise derjenige, der den Schlüssel zuerst anfordert, Knoten A ist derjenige, von dem der Schlüssel angefordert wird, und KTC oder KDC ist das unterstützende Schlüsselzentrum. |
| -- RCVS              | Schlüsselart (KK, *KK, KD oder KDmac, ein temporärer MAC-Schlüssel) eines von NOD empfangenen Schlüssels.   |
| -- FROM              | Teilnehmer, der den empfangenen Schlüssel generiert oder umgewandelt hat; X via Y bedeutet, daß X den Schlüssel über den Knoten Y an NOD weitergibt.  |
| -- UNDR <sup>1</sup> | KEK einfacher oder doppelter Länge, unter dem der Schlüssel empfangen wird; hochgestellte Kleinbuchstaben bezeichnen die Knoten, die den glei-  |

chen KEK besitzen (a ist Knoten A, b ist Knoten B, c ist das Schlüsselzentrum); ein Offset ist implizit; eine Beglaubigung wird durch die Schlüsselform (\*)KNxy dargestellt; "neu" bezeichnet einen neuen KEK für den empfangenen Schlüssel.

- GENS Schlüsselart eines von NOD generierten Schlüssels (KK, \*KK, KD oder KDmac).
- STORES die Form des empfangenen oder generierten Schlüssels, der in NOD gespeichert werden soll; KK//KK bezeichnet die Verdopplung eines Schlüssels KK einfacher Länge zu einem KEK mit Pseudo-Doppellänge; (temp) steht für nur temporäre Speicherung: dieser Schlüssel wird nur für die Verarbeitung von CSMs gespeichert.
- UNDR<sup>2</sup> Schlüssel, unter dem der empfangene oder generierte Schlüssel in NOD gespeichert werden soll; in der Regel in der Form KMx, d.h. Hauptschlüssel von Knoten X (a ist Knoten A, b ist Knoten B, c ist das Schlüsselzentrum); die Steuervektorvariation ist implizit.
- SNDS die Form des generierten oder umgewandelten Schlüssels, der an den durch TO angegebenen Knoten gesendet wird.
- TO Knoten, an den der generierte oder umgewandelte Schlüssel gesendet wird; "X via Y" bedeutet, daß NOD den Schlüssel über Knoten Y an Knoten X sendet; "X und Y" bedeutet, daß NOD den Schlüssel an X und Y sendet (unter verschiedenen KEKs).
- UNDR<sup>3</sup> KEK einfacher oder doppelter Länge, unter dem der generierte oder umgewandelte Schlüssel gesendet wird; ein Offset ist implizit; eine Beglaubigung wird durch die Schlüsselform (\*)KNxy dargestellt; "KK1 und KK2" bedeutet, daß der Schlüssel unter zwei verschiedenen KEKs an die unter TO

aufgeführten Knoten gesendet wird; "neu" bezeichnet einen neuen KEK für den generierten oder umgewandelten Schlüssel.

-- SUP? gibt an, ob die CA-Implementierung von ANSI in NOD diese Option des Szenarios unterstützt.

- Verteilung von \*KK

Der Austausch von Schlüsselchiffrierschlüsseln ist in Fig. 79 in einer Tabelle zusammengefaßt.

- Verteilung von KDs

Der Austausch von Datenschlüsseln ist in Fig. 80 und Fig. 81 in einer Tabelle zusammengefaßt.

- Anweisungssatz zur Unterstützung von ANSI X9.17

- 1. Zur Unterstützung von ANSI X9.17 wurden der CA die folgenden neuen Anweisungen hinzugefügt:

- ANSI Partielle Beglaubigung eines Schlüssels (APNOTR)
- ANSI Umchiffrieren vom Hauptschlüssel (ARFMK)
- ANSI Umchiffrieren unter Hauptschlüssel (ARTMK)
- ANSI Schlüssel umwandeln (AXLTKEY)
- ANSI Datenschlüssel kombinieren (ACOMBKD)

- 2. Zur Unterstützung von ANSI X9.17 sind die folgenden bereits vorhandenen CA-Anweisungen erforderlich:

- Schlüssel generieren  
Es wurde eine Erweiterung der Anweisung "Schlüssel generieren" hinzugefügt, damit Steuervektoren der Art ANSI/KEK verarbeitet werden können.
- Chiffrieren

- Keine Änderung erforderlich.
- Dechiffrieren  
Keine Änderung erforderlich.
- MAC generieren  
Keine Änderung erforderlich.
- MAC überprüfen  
Keine Änderung erforderlich.

#### Überlegungen zum Aufbau

##### - Offset und Beglaubigung

Die Offset-Operation mit einem Schlüssel ist der Prozeß der Umwandlung eines Schlüsselchiffrierschlüssels (KEK) durch EXKLUSIV-ODER-Verknüpfung des Schlüssels mit einem gespeicherten Zählerwert. Die Offset-Funktion wird immer zur Umwandlung eines KEK vor der Chiffrierung eines Schlüssels durch diesen KEK verwendet.

Die Beglaubigung ist ein Verfahren zum Versiegeln von KEKs mit den Identitäten von Sender und beabsichtigtem Empfänger. Ein Beglaubigungsschlüssel KN wird gewöhnlich gebildet, indem ein Beglaubigungssiegel NS als Funktion des KEK, der Identitäten von Sender und Empfänger und dem Offset-Zähler für den betreffenden KEK berechnet wird und NS dann durch EXKLUSIV ODER mit dem KEK verknüpft wird. Da der Zählerwert für einen bestimmten KEK mit der Zeit anwächst, wird NS normalerweise bei jeder gewünschten Beglaubigung von KEK neu berechnet.

Da ein bestimmter ANSI KEK immer genau einem Sender und einem Empfänger zugeordnet ist, ist NS und damit auch KN nur von einer dynamischen Größe abhängig: dem Offset-Zähler von KEK. Es kann daher eine Funktion APNOTR definiert werden, um die statischen Größen des KEK (den Schlüssel selber sowie die Identitäten von Sender und Empfänger) zur Berechnung eines partiellen Beglaubi-



gungsschlüssels KN' zu verwenden. KN' kann dann zusammen mit dem KEK gespeichert und wenn eine Beglaubigung gewünscht wird wieder abgerufen werden, und dann einfach einer Offset-Operation mit dem aktuellen Zählerwert unterzogen werden, um KN zu erzeugen. APNOTR wird im Abschnitt "ANSI Partiellen Beglaubigungsschlüssel erstellen (APNOTR)" beschrieben.

Der Vorteil dieses Ansatzes besteht darin, daß KN nicht jedesmal neu berechnet werden muß, und, was noch wichtiger ist, daß die CA-Funktionen weniger komplex sein können. So müssen in den CA-Funktionen zum Importieren, Umwandeln oder Exportieren von Schlüsseln nicht Beglaubigung und Offset von KEKs unterstützt werden, sondern es muß nur eine Offset-Operation durchgeführt werden. Wenn eine Beglaubigung erforderlich ist, wird die partiell beglaubigte Form des KEK an die betreffende CA-Funktion übergeben, andernfalls der KEK selber. Unabhängig davon führt die CA-Funktion eine Offset-Operation mit dem angegebenen Schlüssel aus, bevor dieser zum Chiffrieren oder Dechiffrieren eines Schlüssels verwendet wird.

#### - ANSI X9.17-Untergruppen

Die in Fig. 79, Fig. 80 und Fig. 81 KK- und KD-Verteilungsszenarien enthalten alle in ANSI X9.17 definierten Optionen. Der mit 'SUP?' betitelten letzten Spalte jeder Tabelle ist zu entnehmen, ob diese Option von der CA unterstützt wird.

Bei den derzeit nicht von der CA unterstützten ANSI-Verteilungsoptionen handelt es sich um solche, die die Generierung von Schlüsselchiffrierschlüsseln einfacher Länge betreffen. In Fig. 79 kann ein ANSI-Knoten in der Punkt-zu-Punkt-Umgebung beispielsweise einen neuen KK mit einfacher Länge generierten und an einen ANSI-Knoten B senden. Dabei stehen vier (\*)KK-Optionen zur Verfügung: KKab, \*KKab, KNab oder \*KNab. Die Generierung von KKs einfacher Länge wird von der CA jedoch nicht unter-

stützt. In der Spalte 'SUP?' ist für die vier ANSI-Verteilungsoptionen deshalb 'Nein' eingetragen.

Aus Gründen der Kompatibilität mit Nicht-CA-Knoten unterstützt die CA den Import neuer KKS einfacher Länge. In Fig. 79 kann z.B. ein ANSI-Knoten B einen KK einfacher Länge unter vier Formen eines anderen (\*)KK von einem ANSI-Knoten empfangen: KKab, \*KKab, KNab oder \*KNab. In der Spalte 'SUP?' steht bei allen Optionen 'JA', was bedeutet daß sie unterstützt werden. Es ist zu beachten, daß importierte KKS einfacher Länge immer in verdoppelter Form gespeichert werden (d.h. 128 Bits in der Form KK//KK).

Die Einschränkung bei KKS einfacher Länge hat auch Auswirkungen auf die KD-Verteilungsoptionen. ANSI X9.17 verlangt, daß bei der Verteilung eines neuen (\*)KK mit einem oder zwei KDs die betreffenden KDs unter dem neuen (\*)KK chiffriert werden. Da die CA die Generierung von KKS einfacher Länge nicht unterstützt, wird auch die Option zur Verteilung neuer KDs unter einem neuen KK einfacher Länge nicht unterstützt. Ein Beispiel hierfür ist in Fig. 80 bei der Punkt-zu-Punkt-Umgebung für Knoten A dargestellt. Aus dieser Tabelle ist ersichtlich, daß die KD-Verteilung unter einem neuen KKab einfacher Länge nicht unterstützt wird.

#### ICV-Verwaltung

ANSI X9.17 unterstützt die Verteilung 64 Bit langer einleitender Kettungswerte (ICVs oder IVs) in chiffrierter oder unchiffrierter Form. Es wird ein CA-Makro GENIV definiert, das die Generierung und Speicherung von ICVs unterstützt. ICVs werden zusammen mit den KDs, für die sie benutzt werden sollen, im Schlüsselspeicher gespeichert. GENIV verwendet die CF-Anweisung KGEN zum Generieren einer 64 Bit langen unchiffrierten Zufallszahl RN ohne Paritätsanpassung. GENIV speichert RN und eine ICV-

\_Modus-Markierung im Schlüsselspeichereintrag für KD. Die ICV\_Modus-Markierung gibt an, ob RN als chiffrierter ICV oder als unchiffrierter ICV interpretiert werden soll. Legt die Markierung fest, daß es sich um einen chiffrierten ICV handelt, muß RN vor einer Verwendung in den Funktionen "Chiffrieren" oder "Dechiffrieren" mit Hilfe des Schlüssels KD dechiffriert werden. Zur Verteilung des generierten ICV und einer ähnlichen Markierung, die besagt, ob der ICV chiffriert oder nicht chiffriert ist, werden die ANSI X9.17-Meldungen RTR und KSM verwendet. GENIV wird außerdem vom Empfänger zur Speicherung des empfangenen ICV im Schlüsselspeicher benutzt. Der Empfänger muß selbstverständlich den ICV\_Modus so angeben wie durch die Markierung in der RTR- oder KSM-Meldung vorgegeben.

#### Akronyme und Abkürzungen

CC	Bedingungscode
CA	Chiffrierarchitektur
CV	Steuervektor (kein Zusammenhang mit ICV oder OCV)
CBC	Chiffreblockkettung. Ein Chiffriermodus des Datenverschlüsselungsstandards.
DED	Dechiffrieren, Chiffrieren und wieder Dechiffrieren
DEA	Datenverschlüsselungsalgorithmus
DES	Datenverschlüsselungsstandard
ECB	Elektronisches Codebuch. Ein Chiffriermodus des DES.
EDE	Chiffrieren, Dechiffrieren und wieder Chiffrieren
ICV	Eingabe-Kettungswert (kein Zusammenhang mit CV)
KDx	Datenschlüssel (x = ganzzahliger Wert)
KEKx	Schlüsselchiffrierschlüssel (x = ganzzahliger Wert)
KM	Hauptschlüssel
KMN	Neuer Hauptschlüssel
KMO	Alter Hauptschlüssel
KPEx	PIN-Chiffrierschlüssel (x = ganzzahliger Wert)
KPGx	PIN-Generierungsschlüssel (x = ganzzahliger Wert)
KPVx	PIN-Überprüfungsschlüssel (x = ganzzahliger Wert)
KKNI	Unmittelbarer beglaubigter Schlüssel, 128 Bit

KKNIL Linke 64 Bit von KKNI  
KKNIR Rechte 64 Bit von KKNI  
KMac Temporärer MAC-Schlüssel für ANSI-Meldung  
MAC Meldungs-Identifikationsüberprüfungscode  
MDC Änderungserkennungscode  
OCV Ausgabe-Kettungswert  
PIN Persönliche Identifikationszahl (in Verbindung mit  
ATMs verwendet)

### Algorithmen

#### - Codierungs- und Decodierungsalgorithmen

Die Anweisungen "Chiffrieren" und "Dechiffrieren" verwenden den ECB-Modus (elektronisches Codebuch) des DES. In diesem Modus gibt es keine Kettung und kein Feedback. Die Funktionsweise des ECB-Modus bei der Chiffrierung und Dechiffrierung ist in Fig. 82 dargestellt.

#### - Verschlüsselungsalgorithmus

Der Chiffrier-/Dechiffrieralgorithmus entspricht dem Datenverschlüsselungsstandard (DES) des National Bureau of Standards bzw. dem Datenverschlüsselungsalgorithmus (DEA) X9.92-1981 des American National Standards Institute (ANSI). Die Chiffreblockkettung (CBC) erfolgt nach den Vorgaben der Verschlüsselungsmodi gemäß ANSI X9.106-1983. Der CBC-Modus der Chiffrieroperation und der Dechiffrieroperation ist in Fig. 83 bzw. Fig. 84 dargestellt.

#### - Meldungs-Identifikationsüberprüfung (MAC)

In Referenz (6), ANSI X9.9 -1986 "American National Standard for Financial Institution Message Authentication (Wholesale)" wird ein Prozeß zur Identifikationsüberprüfung der Meldungen von ei-

nem Urheber an einen Empfänger definiert. Dieser Prozeß ist nicht von Übertragungsmedien und Zahlungssystemen abhängig.

Der Identifikationsüberprüfungsprozeß umfaßt die Berechnung, Übertragung und Überprüfung eines Meldungs-Identifikationsüberprüfungscode (MAC).  $MAC = C$  basiert entweder auf dem vollständigen Meldungstext oder aus ausgewählten Meldungstextelementen. Der MAC wird vom Urheber der Meldung hinzugefügt und an den Empfänger übermittelt. Die Meldung oder die Meldungselemente werden vom Empfänger als echt akzeptiert, wenn der gleiche Algorithmus und der gleiche geheime Schlüssel einen mit dem empfangenen MAC identischen MAC ergeben. Die Sicherheit des Überprüfungsprozesses hängt direkt von der Sicherheit des geheimen Schlüssels ab.

Die MAC-Generierung ist in Fig. 85 dargestellt. Der in diesem Standard beschriebene Algorithmus zur Identifikationsüberprüfung kann entweder mittels des 64-Bit-CBC-Betriebsmodus oder mittels des 64-Bit-CFB-Betriebsmodus gemäß ANSI X3.106-1983 implementiert werden. Beide Modi müssen so initialisiert werden, daß sie äquivalente MACs erzeugen. KEY ist ein 64-Bit-Schlüssel, und A1 bis An sind 64-Bit-Datenblöcke. Bei diesem Standard und CBC-Betriebsmodus muß als einleitender Kettungswert '0' implementiert werden wie in Fig. 85 dargestellt. Ist An kleiner als 64 Bit, werden rechts von den übrigen Bits Nullen angehängt (aufgefüllt). Die linken 32 Bit von (on) werden als MAC verwendet.

HINWEIS: Die Fähigkeit, 48 bis 64 Bit lange MACs zu generieren und zu verarbeiten ist wünschenswert. In diesen Fällen werden die linken 48 Bit oder die gesamte Endausgabe (On) als MAC verwendet.

Der Algorithmus beschreibt die MAC-Generierung für Binärdaten. Die Meldungs-Identifikationsüberprüfung für "Codierte Zeichensätze" ist gemäß ANSI X9.9-1986 zu implementieren; der MAC-Algo-

rithmus wird nach Darstellung der Zeichen als Binärdaten aufgerufen.

- MDC-Algorithmus

Es gibt zwei MDC-Algorithmen:

- 1. MDC\_2 - Zwei Chiffrierungen pro 8-Byte-Eingabedatenblock
- 2. MDC\_4 - Vier Chiffrierungen pro 8-Byte-Eingabedatenblock

Zwei verschiedene Algorithmen geben dem Aufrufenden die Möglichkeit einer 50-prozentigen Leistungssteigerung bei nur geringen von der Anwendung abhängigen Sicherheitseinbußen.

- MDC\_2 (Text)

- 1. Eingabetext mit X'FF' auf ein Vielfaches von 8 Byte auffüllen.
- 2. Den Eingabetext in [n] 8-Byte-Blöcke T8[1] bis T8[n] aufteilen.
- 3. Wenn n = 1 ist, dann n = 2 und T8[2] = 8 Bytes mit dem Wert x'00' setzen.
- 4. Anfangswerte von KD1 und KD2 setzen (siehe unten).
- 5. Für [i] = 1,2,...,n:
  - a. MDCOP(KD1,KD2,T8[i],T8[i])
  - b. KD1 := OUT1
  - c. KD2 := OUT2
  - d. Ende der Schleife
- 6. Die Ausgabe von MDC\_2 ist dann der 16-Byte-MDC := (KD1 // KD2).

- MDC\_4 (Text)

- 1. Eingabetext mit X'FF' auf ein Vielfaches von 8 Byte auffüllen.
- 2. Den Eingabetext in [n] 8-Byte-Blöcke T8[1] bis T8[n] aufteilen.
- 3. Wenn n = 1 ist, dann n = 2 und T8[2] = 8 Bytes mit dem Wert x'00' setzen.
- 4. Anfangswerte von KD1 und KD2 setzen (siehe unten).
- 5. Für [i] = 1,2,...,n:
  - a. MDCOP(KD1,KD2,T8[i],T8[i])
  - b. KD1int := OUT1
  - c. KD2int := OUT2
  - d. MDCOP(KD1int,KD2int,KD2,KD1)
  - e. KD1 := OUT1
  - f. KD2 := OUT2
  - g. Ende der Schleife
- 6. Die Ausgabe von MDC\_4 ist dann der 16-Byte-MDC := (KD1 // KD2).

Die Anfangswerte von KD1 und KD2 lauten:

- 1. KD1 := X'5252525252525252'
- 2. KD2 := X'2525252525252525'

#### - Beglaubigungsalgorithmen

-- Mit KK

Angenommen, KK sei der zur Berechnung des Beglaubigungsschlüssels verwendete Schlüssel. Dann ist:

KKR = KK + FM1 (+ steht für EXKLUSIV ODER, FM1 sind die ersten 8 Bytes der Senderidentifikation)

KKL = KK + TO1 (TO1 sind die ersten 8 Bytes der Identifikation des Empfängers)

NS1 = eKKR(TO2) (TO2 sind die zweiten 8 Bytes der Identifikation des Empfängers)  
NSr = eKKL(FM2) (FM2 sind die zweiten 8 Bytes der Senderidentifikation)  
NS = (linke 32 Bit von NS1) // (rechte 32 Bit von NSr) + CT (CT ist ein 64-Bit-Zähler für KK)  
KN = KK + NS  
KN ist ein beglaubigter Schlüssel zur Chiffrierung eines KD oder eines KK.

-- Mit \*KK

Angenommen, \*KK sei der zur Berechnung des Beglaubigungsschlüssels verwendete Schlüssel. Dann ist:

\*KK = KKl // KKr  
KKR = KKr + FM1 (+ steht für EXKLUSIV ODER, FM1 sind die ersten 8 Bytes der Senderidentifikation)  
KKL = KKl + TO1 (TO1 sind die ersten 8 Bytes der Identifikation des Empfängers)  
NS1 = eKKR(TO2) + CT (TO2 sind die zweiten 8 Bytes der Identifikation des Empfängers und CT ist ein 64-Bit-Zähler für \*KK)  
NSr = eKKL(FM2) + CT (FM2 sind die zweiten 8 Bytes der Senderidentifikation)  
\*KN = (KKl + NS1) // (KKr + NSr)  
\*KN ist ein beglaubigter Schlüssel zur Chiffrierung eines KD oder eines (\*)KK.

### Standards und Definitionen

#### - Standards

ANSI X2.92 - 1981 "Data Encryption Algorithm".

ANSI X9.106 - 1983 "Modes of DEA Operation".

MA9-88-011



- ANSI X9.2 - 198X    "Interchange Message Specification for Debit and Credit Card Message Exchange Among Financial Institutions". Dieser Standard definiert eine gemeinsame Schnittstelle, über die von einer Bankkarte ausgehende Meldungen bezüglich einer finanziellen Transaktion zwischen privaten Systemen ausgetauscht werden können. Er definiert Struktur, Format und Inhalt der Meldungen sowie Datenelemente und Werte für Datenelemente.
- ANSI X9.8 - 1982    "American National Standard for Personal Identification Number (PIN) Management and Security". Dieser Standard legt Standards und Richtlinien für die Verwaltung und Sicherheit der Lebensdauer von persönlichen Identifikationsnummern (PINs) fest.
- ANSI X9.9 - 1986    "American National Standard für Financial Institution Message Authentication (Wholesale)". Dieser Standard definiert ein Verfahren zur Identifikationsüberprüfung von Finanzmeldungen (in Massen), einschließlich Geldtransfers (z.B. telegraphisch), Akkreditiven, Transfers von Wertpapieren, Kreditvereinbarungen und Devisenverträgen.
- ANSI X9.17 -1985    "Financial Institution Key Management (Wholesale)". Dieser Standard definiert Verfahren (einschließlich eines Protokolls) zur Generierung, Übertragung und Verwendung von Chiffrierschlüsseln für die Identifikationsüberprüfung und Chiffrierung.
- ANSI X9.19 - 198X    "Financial Institution Retail Message Authentication". Dieser Standard definiert ein Verfahren zur Identifikationsüberprüfung von Finanzmeldungen für Transaktionen in geringen Mengen.

- ANSI X9.23 - 198X "Encryption of Wholesale Financial Messages". Dieser Standard definiert ein Verfahren zur Chiffrierung großer Mengen von Finanzmeldungen, das die Vertraulichkeit gewährleistet (z.B. bei telegrafischen Transfers, Akkreditiven usw.).
- ISO DIS 8583 "Bank Card Originated Messages - Interchange Message Specifications - Content for Financial Transactions". Dieser internationale Standard definiert eine gemeinsame Schnittstelle, über die von einer Bankkarte ausgehende Meldungen bezüglich einer finanziellen Transaktion zwischen privaten Systemen ausgetauscht werden können. Er definiert Struktur, Format und Inhalt der Meldungen sowie Datenelemente und ihre Werte.
- ISO DIS 8720 "Message Authentication"
- ISO DP 8730 "Banking - Requirements for Standard Message Authentication (Wholesale)". Dieser internationale Standard definiert ein Verfahren zur Identifikationsüberprüfung von Meldungen, die zwischen Geldinstituten ausgetauscht werden, mittels eines Meldungs-Identifikationsüberprüfungs-codes (MAC).
- ISO DP 8731 "Banking - Approved Algorithms for Message Authentication - Part 1: DES-1 Algorithm". In diesem Teil von ISO 8731 geht es um den Datenverschlüsselungsalgorithmus (DEA-1) als Verfahren zur Verwendung bei der Berechnung des Meldungs-Identifikationsüberprüfungs-codes (MAC). Teil 2 betrifft Nicht-DEA-Algorithmen.
- ISO DP 8732 "Banking - Key Management Wholesale". Dieser internationale Standard definiert Verfahren zur Verwaltung von Schlüsselmaterial für die Chiffrierung und Identifikationsüberprüfung

ISO DP 9546

von Meldungen, die im Rahmen von Massenfinanztransaktionen ausgetauscht werden. "Personal Identification Number Management and Security Part 1 - PIN Protection Principles and Technique". Dieser Standard definiert das Minimum an Sicherheitsvorkehrungen, die für eine effektive Verwaltung von PINs erforderlich sind. Es werden Standardmittel für den Austausch von PIN-Daten beschrieben.

- Übersichtstabellen zu den Anweisungen und Makros

In Fig. 86, Fig. 87 und Fig. 88 sind die Gleichungen für die einzelnen CA-Anweisungen zusammengefaßt.

- Referenzliteratur

1. GC31-2070-0: 4700 Finance Communication System, Controller Programming Library, Band 5, Cryptographic Programming
2. ANSI for Information Systems - DEA - Modes of Operations  
ANSI X3.106-1983
3. ANSI Data Encryption Algorithm - ANSI X3.92-1981
4. Financial Institution Message Authentication (Wholesale)  
ANSI X9.9-1986
5. Financial Institution Key Management (Wholesale)  
ANSI X9.17-1985
6. R. C. Summers, Systems Journal, Band 23, Nr. 4, 1984, S. 309 - 325, "An Overview of Computer Security"
7. W. L. Price, National Physical Laboratory Technical Memo DITC 4/86, Januar 1986, "Physical Security of Transaction Devices"
8. A. E. Winblad (Sandia National Laboratories), SAND85-0766C, Dezember 1985, "Future Developments in Physical Protection Against the Insider Threat"

9. John Horgan, IEEE Spectrum, Juli 1985, S. 30 - 41, "Thwarting the Information Thieves"
10. T. H. DiStefano, NBS Special Publication 400-23, ARPA/NBS Workshop IV, Gaithersburg, MD, erschienen im März 1976, "Photoemission and Photovoltaic Imaging of Semiconductor Surfaces"
11. D. C. Shaver et al., IEEE Electron Device Letters, Band. EDL-4, Nr. 5, Mai 1983, "Electron-Beam Programmable 128K-Bit Wafer-Scale EPROM"
12. A. Gercekci (Motorola), US-Patentschrift 4,394,750, 19. Juli 1983, "PROM Erase Detector"
13. D. J. DiMaria, Proceedings of the International Topical Conference, Yorktown Heights, NY, März 1978, S. Pantelides (Hrsg.), Pergamon Press, "The Physics of SiO<sub>2</sub> and its Interfaces"
14. D. R. Young et al., J. Appl. Phys. 50(10), Oktober 1979, S. 6366-72, "Electron Trapping in SiO<sub>2</sub> at 295 and 77K"
15. R. Singh et al., IEEE Trans. on Nuclear Science, Band NS-31, Nr. 6, Dezember 1984, S. 1518-23, "Total-Dose and Charge-Trapping Effects in Gate Oxides for CMOS LSI Devices"
16. R. Reich, IEEE Electron Device Letters, Band EDL-7, Nr. 4, April 1986, S. 235 - 237, "Radiation-Dependent Hot-Carrier Effects"
17. David Chaum (Hrsg.), Advances in Cryptology, Proceedings of Crypto 83, Plenum Press, NY, 1984, "Design Concepts for Tamper Responding Systems"

## A N S P R Ü C H E

1. Ein Verfahren zur Überprüfung, daß für einen Chiffrierschlüssel angeforderte Schlüsselverwaltungsfunktionen vom Urheber des Schlüssels in einem Datenverarbeitungssystem autorisiert worden sind, das Chiffrierserviceanforderungen zur Verwaltung von Chiffrierschlüsseln verarbeitet, denen Steuervektoren zugeordnet sind, welche die Funktionen, zu denen jeder Schlüssel von seinem Urheber ermächtigt wurde, definieren, wobei das Verfahren folgende Schritte umfaßt:

Empfangen einer Chiffrierserviceanforderung zur Ausführung einer Schlüsselverwaltungsfunktion an einem Chiffrierschlüssel in einer Chiffriervorrichtung 4 mit einer sicheren Grenze 6, durch die ein Eingangspfad und ein Ausgangspfad 8 verläuft;

Empfangen eines dem Chiffrierschlüssel zugeordneten Steuervektors (Fig. 10) und Prüfung, ob der Steuervektor zu der durch die Chiffrierserviceanforderung angeforderte Schlüsselverwaltungsfunktion berechtigt; sowie

Signalisieren 20, daß die Schlüsselverwaltungsfunktion autorisiert ist, und Starten der angeforderten Schlüsselverwaltungsfunktion mit dem Chiffrierschlüssel.

2. Ein Verfahren gemäß Anspruch 1, das außerdem folgende Schritte umfaßt:

Speichern des Chiffrierschlüssels in verschlüsselter Form in einer Speichervorrichtung 22, wobei der Chiffrierschlüssel unter einem Speicherschlüssel verschlüsselt wird, der ein logisches Produkt des zugeordneten Steuervektors und eines Hauptschlüssels 18 ist.

3. Ein Verfahren gemäß Anspruch 2, bei dem außerdem

die Chiffrierserviceanforderung die Rekonstruktion des Chiffrierschlüssels aus der Speichervorrichtung betrifft;

die verschlüsselte Form des Chiffrierschlüssels aus der Speichervorrichtung empfangen und unter dem Speicherschlüssel, der ein logisches Produkt des zugeordneten Steuervektors und des Hauptschlüssels ist, dechiffriert wird.

4. Ein Verfahren gemäß den vorausgehenden Ansprüchen, bei dem der Speicherschlüssel das Exklusiv-ODER-Produkt des zugeordneten Steuervektors und des Hauptschlüssels ist.
5. Ein Verfahren gemäß den vorausgehenden Ansprüchen, bei dem der zugeordnete Steuervektor mit der verschlüsselten Form des Chiffrierschlüssels in der Speichervorrichtung gespeichert wird.
6. Ein Verfahren gemäß den vorausgehenden Ansprüchen, bei dem der zugeordnete Steuervektor Felder enthält, in denen automatisierte Arten von Chiffrierfunktionen einschließlich Schlüsselverwaltungsfunktionen, Datenverschlüsselungs- bzw. Datenentschlüsselungsfunktionen und PIN-Verarbeitungsfunktionen definiert sind, und bei dem die Art der Schlüsselverwaltungsfunktionen durch folgende Angaben näher bezeichnet wird:
- Exportkontrolle und Verwendung;
  - Verbindungssteuerung, die angibt, ob ein einem Chiffrierschlüssel zugeordneter Steuervektor bei der Übertragung vom Datenverarbeitungssystem an ein angeschlossenes fernes Datenverarbeitungssystem aufgrund der Merkmale des fernen Datenverarbeitungssystems übergangen werden kann;

- Angabe, ob der zugeordnete Schlüssel von einem ANSI-Datenverarbeitungssystem verarbeitet werden kann.
7. Ein Verfahren gemäß den vorausgehenden Ansprüchen, bei dem der zugeordnete Steuervektor Felder enthält, in denen die Trennung von Schlüsselchiffrierschlüsseln auf der Basis zweier sich gegenseitig ausschließender Verwendungszwecke erzwungen wird.
8. Ein Verfahren gemäß Anspruch 7, bei dem die sich gegenseitig ausschließenden Verwendungszwecke folgender Liste entnommen sind:
- für beglaubigte und nicht beglaubigte Schlüssel;
  - für eine erste Art von Schlüsselchiffrierschlüsseln mit Steuervektoren und eine zweite Art von Schlüsselchiffrierschlüsseln ohne Steuervektoren;
  - für eine nur von Sendern verwendete erste Art von Schlüsselchiffrierschlüsseln und eine nur von Empfängern verwendete zweite Art von Schlüsselchiffrierschlüsseln;
  - für eine erste Art von Schlüsselchiffrierschlüsseln, die für die Umwandlung von Schlüsseln für die Versendung ohne die Möglichkeit zum Export vorhandener unter einem Hauptschlüssel gespeicherter Datenschlüssel verwendet wird, und für eine zweite Art von Schlüsselchiffrierschlüsseln, die für die Umwandlung von Schlüsseln für die Versendung mit der Möglichkeit zum Export vorhandener unter einem Hauptschlüssel gespeicherter Datenschlüssel verwendet wird;
  - für eine erste Art von Schlüsselchiffrierschlüsseln, die nur für den Export generiert werden können, und eine zweite Art von Schlüsselchiffrierschlüsseln, die für eine lokale Verwendung und für den Export generiert werden können;
  - für eine erste Art von Schlüsselchiffrierschlüsseln, die bei der Umwandlung ohne die Möglichkeit zur lokalen Ver-

wendung benutzt werden können, und eine zweite Art von Schlüsselchiffrierschlüsseln, die bei der Umwandlung benutzt werden können, und bei denen eine lokale Verwendung zulässig ist;

- für eine erste Art von Schlüsselchiffrierschlüsseln, die bei der Wiederverschlüsselung von Hauptschlüsseloperationen verwendet werden können, und eine zweite Art von Schlüsselchiffrierschlüsseln, die nicht für die Wiederverschlüsselung von Hauptschlüsseloperationen verwendet werden können.

9. Ein Verfahren gemäß den vorausgehenden Ansprüchen, bei dem der zugeordnete Steuervektor ein Feld enthält, in dem angegeben ist, ob der Chiffrierschlüssel einfache oder doppelte Länge besitzt.

10. Ein Verfahren gemäß den vorausgehenden Ansprüchen zur Ausführung einer Schlüsselgruppengenerierungsfunktion, das außerdem folgende Schritte umfaßt:

Erzeugung einer Zufallszahl für den Verschlüsselungsprozeß;

Empfangen einer Chiffrierserviceanforderung über den Eingangspfad, um mit den ersten und zweiten Steuervektoren C3 und C4 aus der Zufallszahl ein Schlüsselpaar zu generieren und als Antwort darauf ein Berechtigungssignal an den Chiffrierprozeß zu senden, das besagt, daß die Funktion zur Generierung eines Schlüsselpaares aus der Zufallszahl berechtigt ist;

als Antwort auf das Berechtigungssignal Ausgabe der Zufallszahl als erster generierter Schlüssel in einer chiffrierten Form, in der die Zufallszahl unter einem Schlüssel chiffriert wird, der ein logisches Produkt des ersten zugeordneten Steuervektors C3 und eines ersten Schlüssels K1 ist;



als Antwort auf das Berechtigungssignal Ausgabe der Zufallszahl als zweiter generierter Schlüssel in einer chiffrierten Form, in der die Zufallszahl unter einem Schlüssel chiffriert wird, der ein logisches Produkt des zweiten zugeordneten Steuervektors C4 und eines zweiten Schlüssels K2 ist.

11. Ein Verfahren gemäß Anspruch 10 zur Erzeugung zweier Schlüssel, die im Datenverarbeitungssystem verwendbar sind, wobei der erste Schlüssel K1 und der zweite Schlüssel K2 den Hauptschlüssel bilden, der die Verwendung im lokalen Datenverarbeitungssystem ermöglicht.
12. Ein Verfahren gemäß Anspruch 10 zur Erzeugung eines ersten generierten Schlüssels, der nur in dem Datenverarbeitungssystem benutzt werden kann, das ein lokales Datenverarbeitungssystem ist, und eines zweiten generierten Schlüssels, der an ein an das lokale System angeschlossenes fernes Datenverarbeitungssystem exportiert werden kann, wobei der erste Schlüssel K1 der Hauptschlüssel ist, der die Benutzung im lokalen Datenverarbeitungssystem ermöglicht, und der zweite Schlüssel K2 ein Schlüsselchiffrierschlüssel KEK2 mit einem zugeordneten Steuervektor C2, und der Steuervektor C2 zum Export an das ferne Datenverarbeitungssystem berechtigt.
13. Ein Verfahren gemäß Anspruch 10 zur Erzeugung eines ersten generierten Schlüssels, der an ein erstes an das lokale Datenverarbeitungssystem angeschlossenes fernes Datenverarbeitungssystem exportiert werden kann, und eines zweiten generierten Schlüssels, der an ein zweites an das lokale Datenverarbeitungssystem angeschlossenes fernes Datenverarbeitungssystem exportiert werden kann, wobei der erste Schlüssel K1 ein Schlüsselchiffrierschlüssel KEK1 mit einem zugeordneten Steuervektor C1 ist und der Steuervektor C1 zum Export an das erste ferne Datenverarbeitungssystem berechtigt, und wobei der zweite Schlüssel K2 ein Schlüsselchiff-

rierschlüssel KEK2 mit einem zugeordneten Steuervektor C2 ist und der Steuervektor C2 zum Export an das zweite ferne Datenverarbeitungssystem berechtigt.

14. Ein Verfahren gemäß Anspruch 10 zur Erzeugung eines ersten generierten Schlüssels, der nur in dem Datenverarbeitungssystem benutzt werden kann, das ein lokales Datenverarbeitungssystem ist, und eines zweiten generierten Schlüssels, der von einem an das lokale System angeschlossenen Benutzergerät importiert werden kann, wobei der erste Schlüssel K1 der Hauptschlüssel ist, der ausschließlich zur Benutzung im lokalen Datenverarbeitungssystem berechtigt, und der zweite Schlüssel K2 ein Schlüsselchiffrierschlüssel KEK2 mit einem zugeordneten Steuervektor C2 ist und der Steuervektor C2 zum Import vom Benutzergerät in das lokale System berechtigt.
15. Ein Verfahren gemäß Anspruch 10 zur Erzeugung eines ersten generierten Schlüssels, der von einem an das lokale Datenverarbeitungssystem angeschlossenen Benutzergerät importiert werden kann, und eines zweiten generierten Schlüssels, der an ein an das lokale System angeschlossenes fernes Datenverarbeitungssystem exportiert werden kann, wobei der erste Schlüssel K1 ein Schlüsselchiffrierschlüssel KEK1 mit einem zugeordneten Steuervektor C1 ist und der Steuervektor C1 zum Import vom Benutzergerät berechtigt, und wobei der zweite Schlüssel K2 ein Schlüsselchiffrierschlüssel KEK2 mit einem zugeordneten Steuervektor C2 ist und der Steuervektor C2 zum Export an das zweite ferne Datenverarbeitungssystem berechtigt.
16. Ein Verfahren gemäß den vorausgehenden Ansprüchen zur Durchführung einer Wiederverschlüsselung vom Hauptschlüssel, das außerdem über Mittel zur Prüfung des dem Schlüsselchiffrierschlüssel KEK zugeordneten Steuervektors C1 verfügt, um sicherzustellen, daß die Wiederverschlüsselung vom Haupt-

schlüssel autorisiert ist, und zur Prüfung, ob der dem Schlüssel K zugeordnete Steuervektor C2 zum Export des Schlüssels K anhand der Wiederverschlüsselung vom Hauptschlüssel berechtigt.

17. Ein Verfahren gemäß Anspruch 16, bei dem der zugeordnete Steuervektor C3 dem fernen Prozessor selektiv ermöglicht, den Chiffrierschlüssel K wiederzuexportieren.
18. Ein Verfahren gemäß Anspruch 16, bei dem der dem Schlüsselchiffrierschlüssel KEK zugeordnete Steuervektor C1 geprüft wird, um sicherzustellen, daß die Funktion zur Wiederverschlüsselung zum Hauptschlüssel autorisiert ist.
19. Ein Verfahren gemäß Anspruch 16, bei dem der vom fernen Datenverarbeitungssystem empfangene Steuervektor C3 selektiv einen weiteren Wiederexport des Schlüssels K vom lokalen Datenverarbeitungssystem erlaubt.
20. Ein Verfahren gemäß Anspruch 16, bei dem der Steuervektor C2 vom lokalen Datenverarbeitungssystem selektiv so gesetzt wird, daß ein weiterer Wiederexport des Chiffrierschlüssels K vom lokalen Datenverarbeitungssystem erlaubt ist.
21. Ein Verfahren gemäß den vorausgehenden Ansprüchen zur Durchführung einer Schlüsselumwandlungsfunktion wobei der Steuervektor C1, der dem Importschlüssel-Chiffrierschlüssel KEK1 zugeordnet ist, geprüft wird, um sicherzustellen, daß KEK1 als Importschlüssel-Chiffrierschlüssel bei der Schlüsselumwandlungsfunktion autorisiert ist, und wobei der Steuervektor C2, der dem Exportschlüssel-Chiffrierschlüssel KEK2 zugeordnet ist, geprüft wird, um sicherzustellen, daß KEK2 als Exportschlüssel-Chiffrierschlüssel bei der Schlüsselumwandlungsfunktion autorisiert ist.

22. Ein Verfahren gemäß Anspruch 1 bis 9 zur Durchführung einer Funktion zur Wiederverschlüsselung vom Hauptschlüssel mit einer geringeren Exportberechtigung für den Empfänger, wobei der dem Schlüsselchiffrierschlüssel KEK zugeordnete Steuervektor C1 geprüft wird, um sicherzustellen, daß die Funktion zur Wiederverschlüsselung vom Hauptschlüssel autorisiert ist, und wobei der dem Schlüssel K zugeordnete Steuervektor C2 zum Export des Schlüssels K mittels der Funktion zur Wiederverschlüsselung vom Hauptschlüssel berechtigt.
23. Ein Verfahren gemäß den vorausgehenden Ansprüchen, bei dem der zugeordnete Steuervektor ein Feld enthält, in dem die Verbindungssteuerung definiert wird, die festlegt, ob ein einem Chiffrierschlüssel zugeordneter Steuervektor bei der Übertragung von Datenverarbeitungssystem an ein daran angeschlossenes fernes Datenverarbeitungssystem aufgrund der Merkmale des fernen Datenverarbeitungssystems übergangen werden kann.
24. Ein Verfahren gemäß den vorausgehenden Ansprüchen, bei dem der zugeordnete Steuervektor ein Feld enthält, in dem angegeben ist, ob der Chiffrierschlüssel einfache oder doppelte Länge besitzt.
25. Ein Datenverarbeitungssystem, das Chiffrierserviceanforderungen für die Verwaltung von Chiffrierschlüsseln verarbeitet, denen Steuervektoren zugeordnet sind, welche die Funktionen definieren, zu deren Ausführung die einzelnen Schlüssel von ihrem Urheber ermächtigt wurden, und das folgende Komponenten umfaßt:  
  
eine Chiffriervorrichtung 4 mit einer sicheren Grenze 6, durch die ein Eingangspfad 8 zum Empfang der Chiffrierserviceanforderungen, der Chiffrierschlüssel und der zugeordneten Steuervektoren sowie ein Ausgangspfad 8 für Antworten darauf

verläuft, wobei sich innerhalb der Grenze 6 ein an den Eingangspfad gekoppelter Speicher 10 für Chiffrieranweisungen, ein Mittel 14 zur Prüfung der Steuervektoren und ein an den Anweisungsspeicher gekoppeltes Chiffrierverarbeitungsmittel 16, sowie ein an das Verarbeitungsmittel 16 gekoppelter Hauptschlüsselspeicher 18, der einen sicheren Ort für die Ausführung von Schlüsselverwaltungsfunktionen als Antwort auf die empfangenen Serviceanforderungen bietet, befindet;

wobei der Chiffrieranweisungsspeicher 10 über den Eingangspfad 8 eine Chiffrierserviceanforderung zur Durchführung einer Schlüsselverwaltungsfunktion mit einem Chiffrierschlüssel empfängt; und

das Mittel 14 zur Prüfung der Steuervektoren einen an den Eingangspfad 8 gekoppelten Eingang zum Empfang eines dem Chiffrierschlüssel zugeordneten Steuervektors (Fig. 10) enthält, sowie einen an den Chiffrieranweisungsspeicher 10 gekoppelten Eingang zum Empfang von Steuersignalen zum Starten der Prüfung, ob der Steuervektor zu der von der Chiffrierserviceanforderung angeforderte Schlüsselverwaltungsfunktion berechtigt;

•

das Mittel 14 zur Prüfung der Steuervektoren einen an einen Eingang des Chiffrierverarbeitungsmittels 16 angeschlossenen Autorisierungsausgang 20 besitzt, über den signalisiert wird, daß die Schlüsselverwaltungsfunktion autorisiert ist, deren Empfang durch das Chiffrierverarbeitungsmittel 16 die Ausführung der angeforderten Schlüsselverwaltungsfunktion mit dem Chiffrierschlüssel startet.

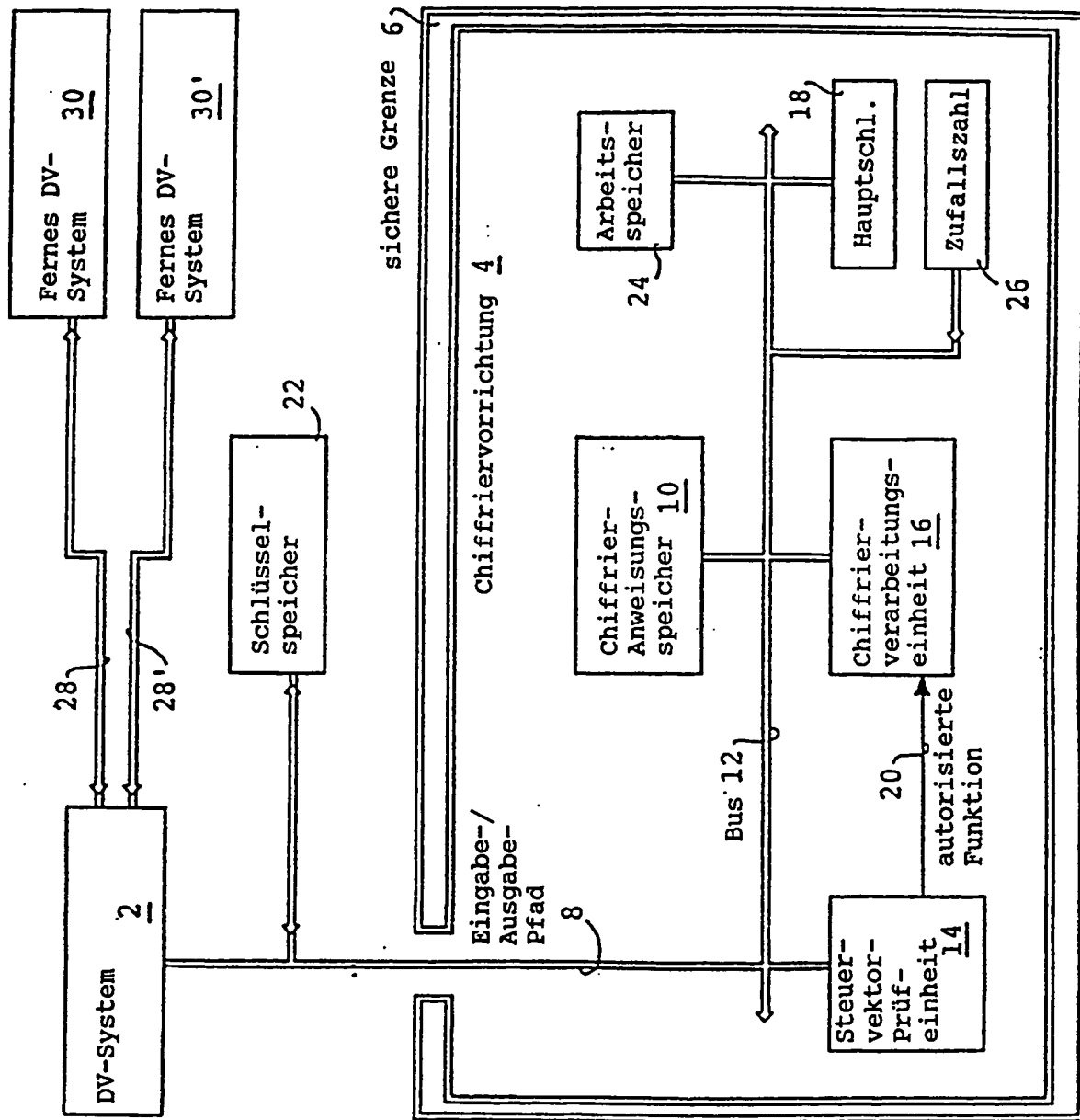


FIG. 1.

FIG. 2.

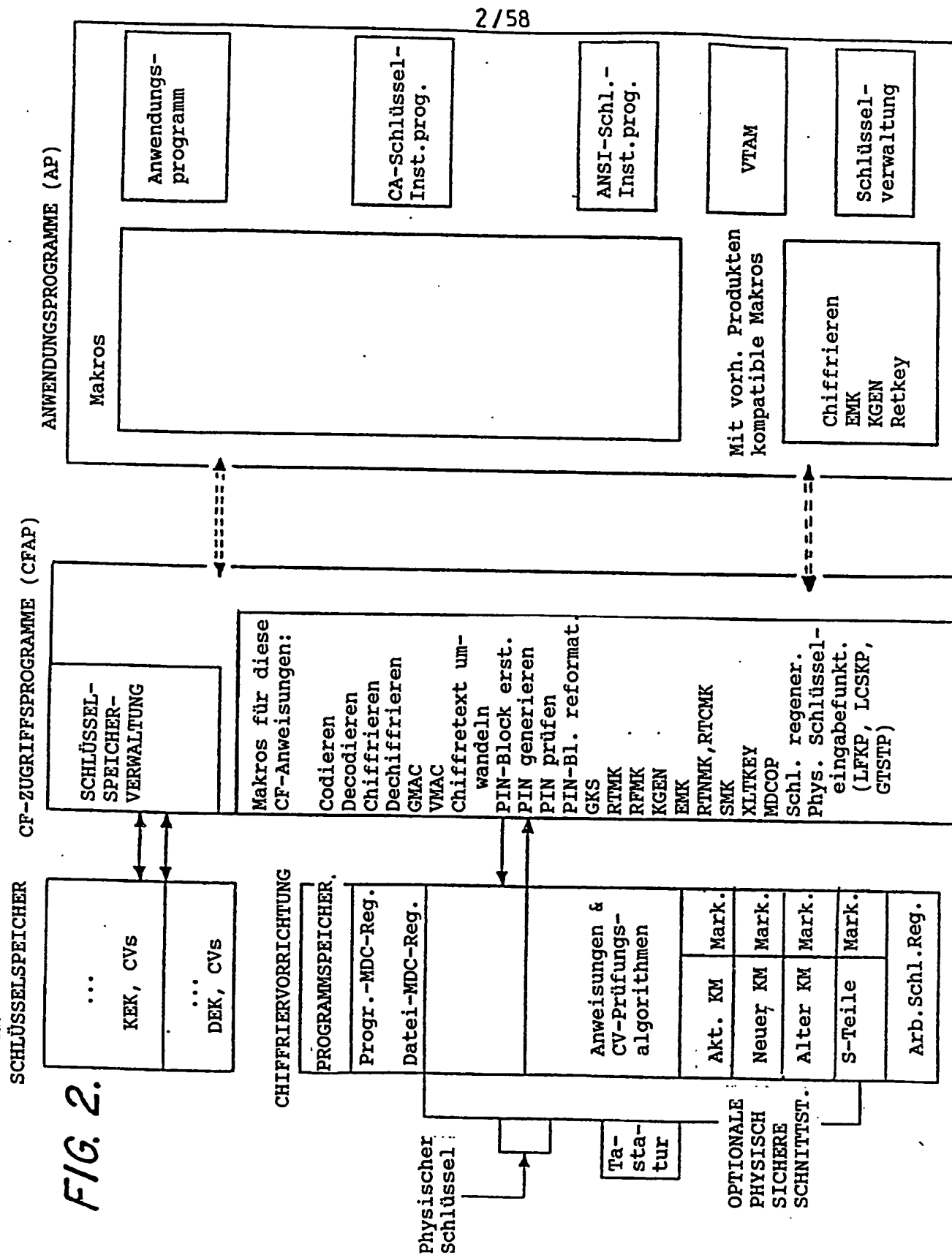
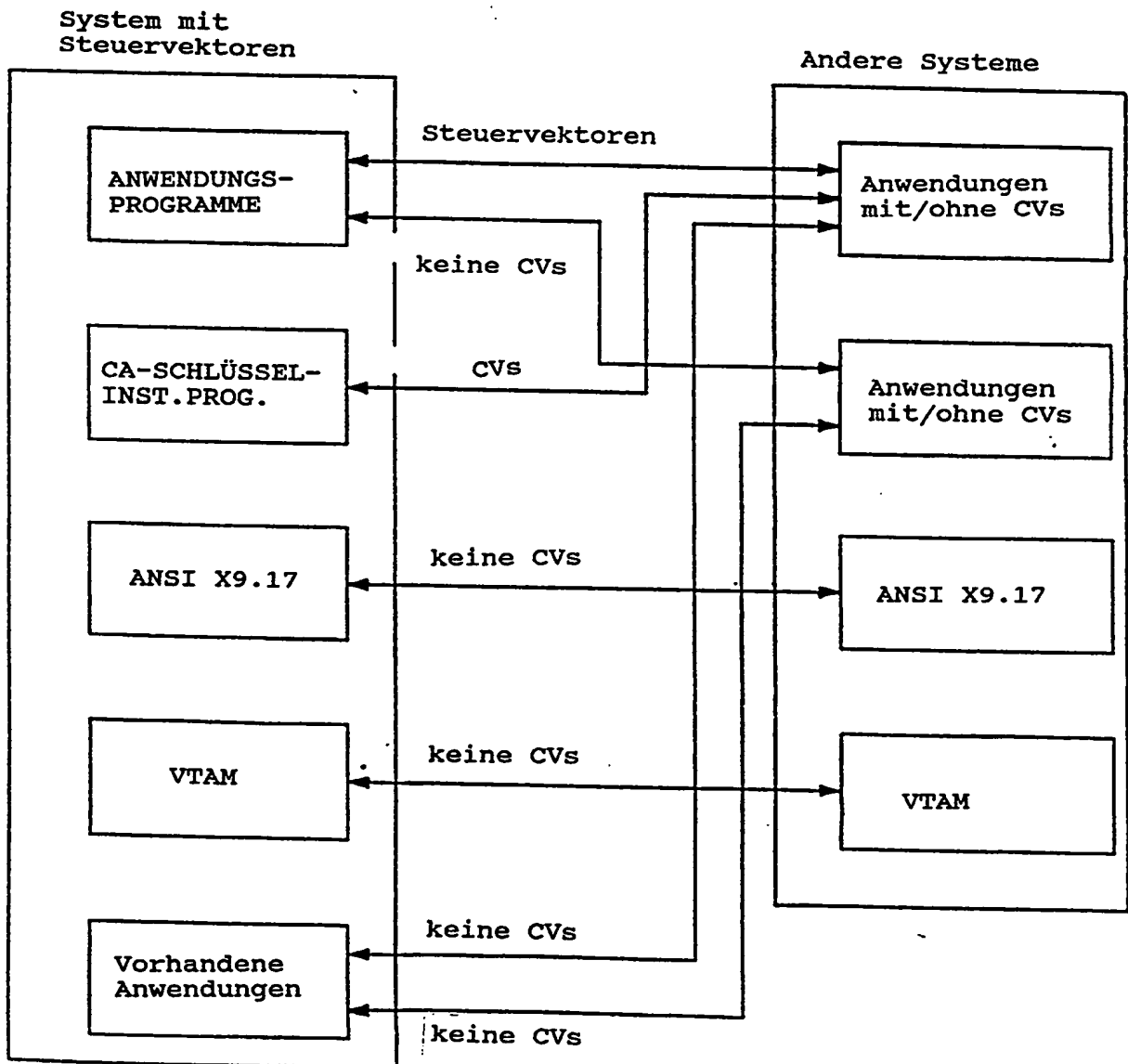


FIG. 3.





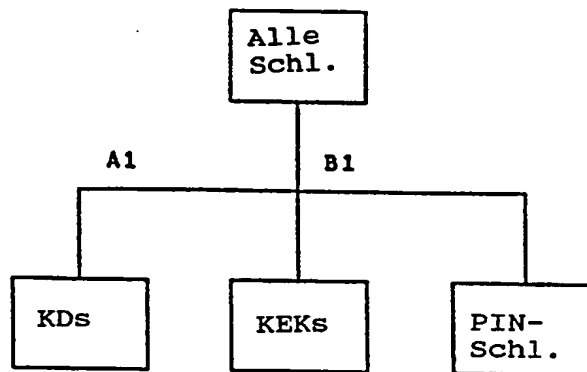
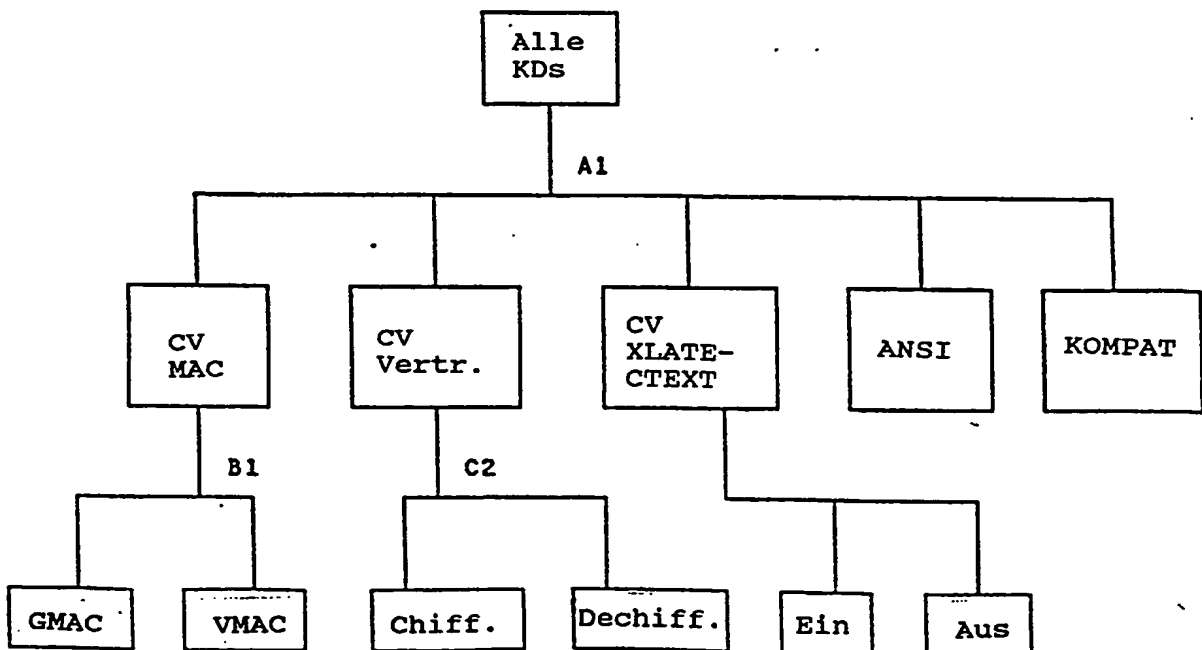
**FIG. 4.****FIG. 5.**

FIG. 6.

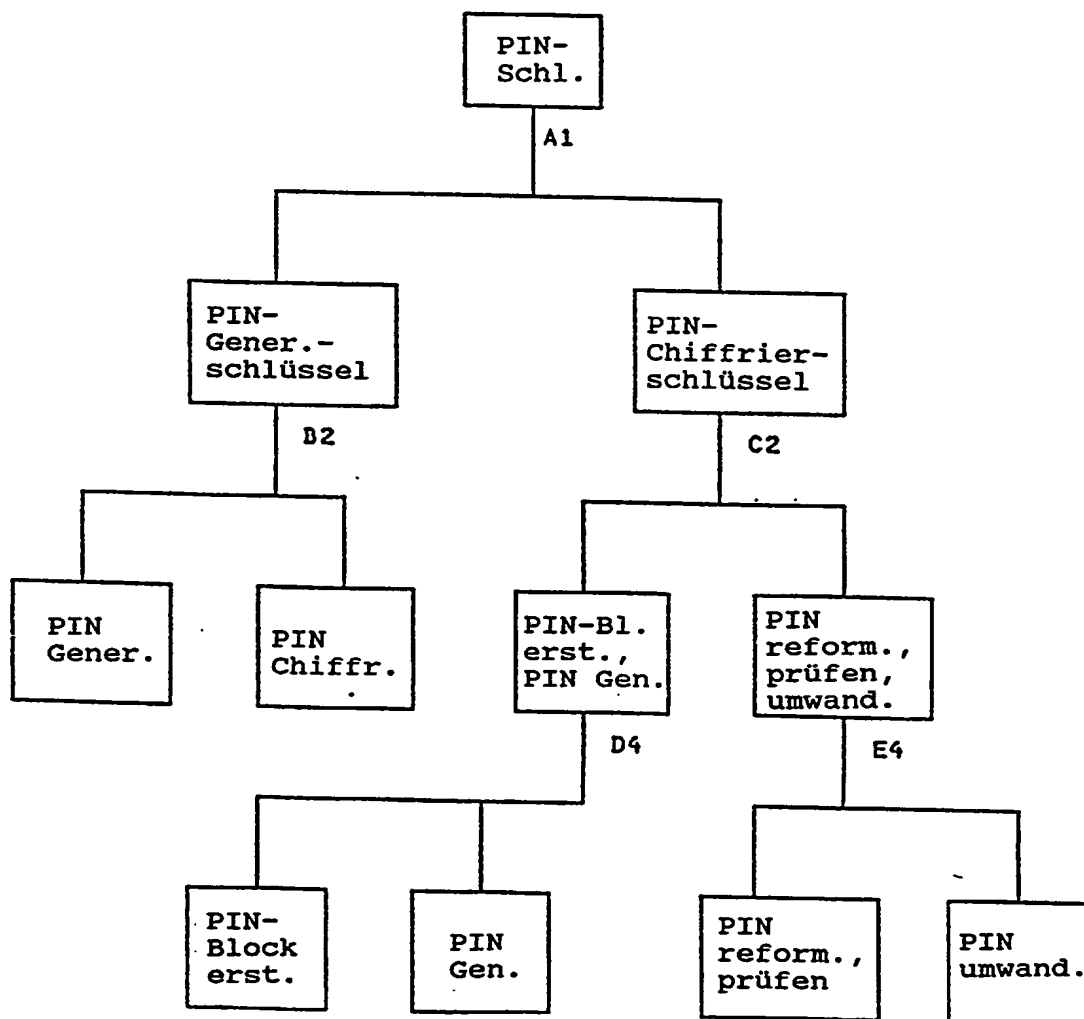


FIG. 7.

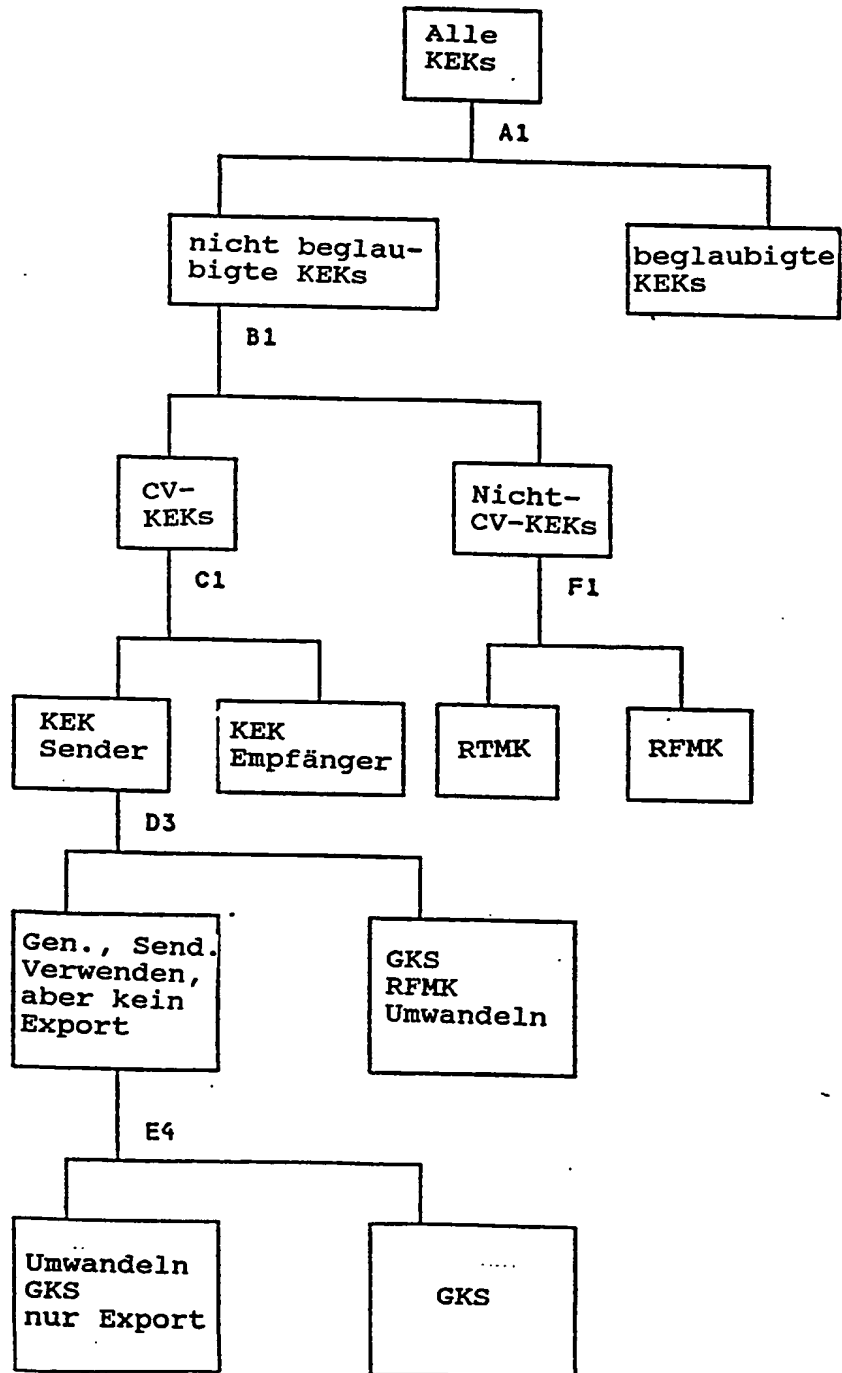
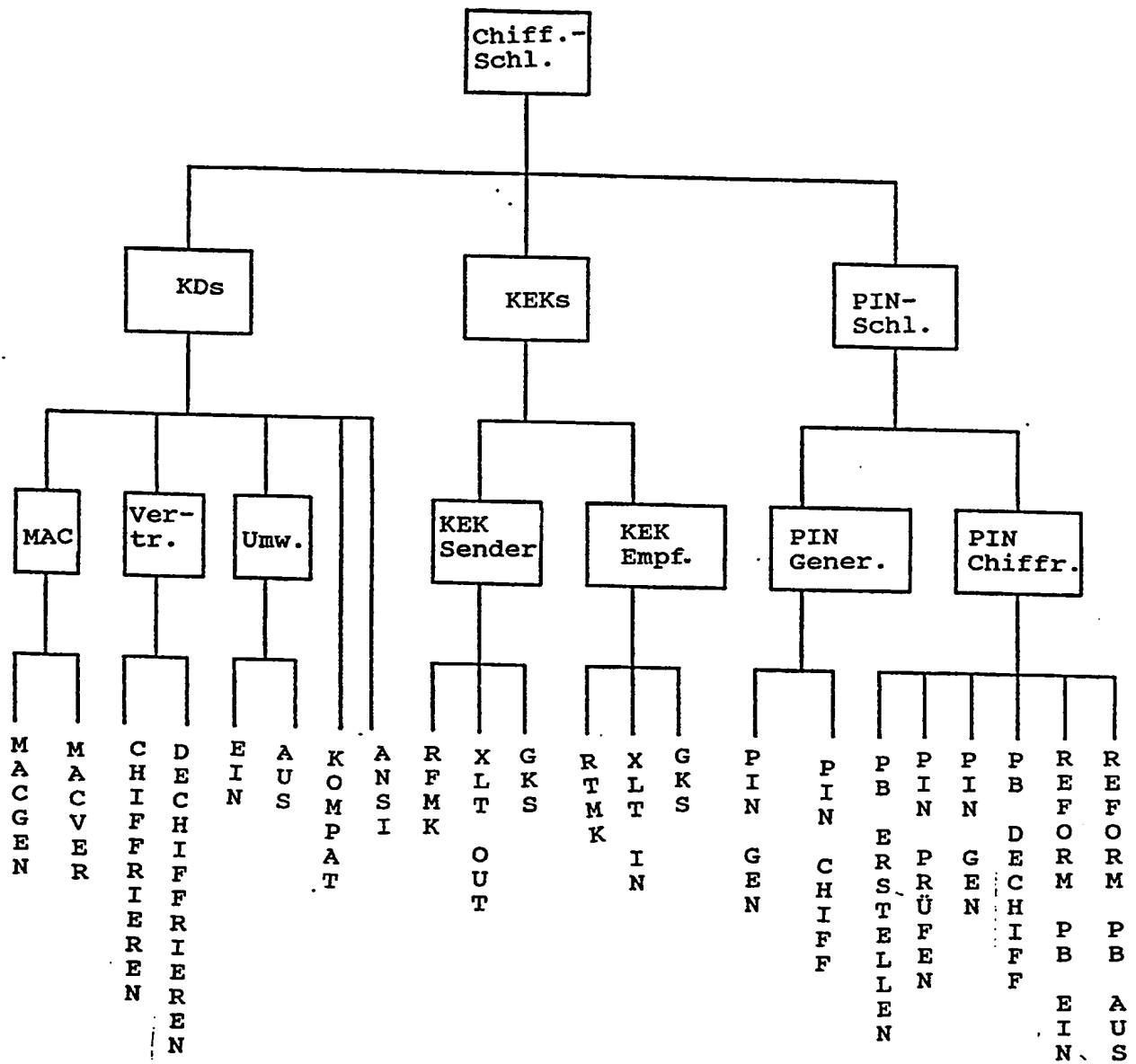


FIG. 8.

Art der Trennung	Priorität			
	1	2	3	4
Elementare Schlüssel				
KD : KEK, PIN-Schlüssel	X			
KEK : PIN-Schlüssel	X			
Datenschlüssel				
Kompatibilität : alle anderen	X			
ANSI : alle anderen	X			
MAC : Vertraulichkeit, Chiffretextumwandlung	X			
Chiffretextumwandlung : Vertraulichkeit	X			
MAC-Generierung : MAC-Prüfung	X			
Chiffrieren : Dechiffrieren	X			
XLT CTEXT(IN) : XLT CTEXT(OUT)		X		
		X		
PIN-Schlüssel				
PIN-Generierung : PIN-Chiffrierung	X			
PIN-Generierungsschlüssel				
PIN-Generierung : PIN-Chiffrierung		X		
PIN-Chiffrierschlüssel				
PIN-Block-Erstellung und PIN-Generierung		X		
: Reformat., Prüfen und Umwandeln von PINs				
PIN-Block-Erstellung: PIN-Generierung				X
Reformat. und Prüfen von PINs : PIN-Umwandlung				X
Schlüsselchiffrierschlüssel				
Beglaubigung : keine Beglaubigung	X			
CV KEKs : KEKs für Nicht-CV-Systeme	X			
Nicht-CV RTMK : Nicht-CV RFMK	X			
KEK Sender : KEK Empfänger	X			
GKS, XLT : RFMK, GKS, XLT			X	
GKS (nur Export), XLT : GKS (allg. Verwendung)			X	

FIG. 9.



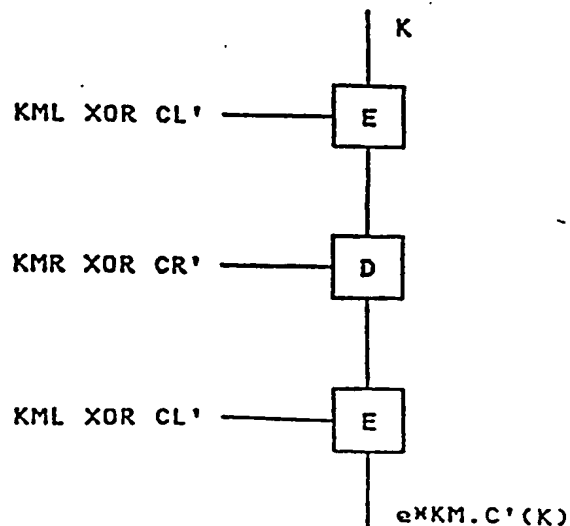
*FIG. 10.*

CV-ART H- Art	U- Art	EXPORT- KONTR.	VERWEN- DUNG	AV	SOFTWARE- BITS	LÄNGE	RESER- VIERT	PARI- TÄT
4b	3b	1b		2b	12b	2b		8b
CF		CF	CF	CF	CFAP	CF	CF	CF

Hinweis: Die Anzahl der Bits in den Feldern VERWENDUNG und RESERVIERT variiert je nach CV-ART des Steuervektors.

*FIG. 11.*

CV-ART DAT. VER- SCH. TR.	EXPORT- KONTR.	VERWENDUNG E D	AV	SOFTWARE CV Verw Version	LÄNGE	RESER- VIERT	PARI- TÄT
4b 3b	1b	1b 1b	2b	6b 6b	2b	30b	8b
CF	CF	CF	CF	CFAP	CF	CF	CF

*FIG. 12.*

*FIG. 13.*

CV-ART DAT. MAC SCH.	EXPORT- KONTR.	VERWENDUNG MG MV	AV	SOFTWARE CV Verw. Version	LÄNGE	RESER- VIERT	PARI- TÄT
4b 3b	1b	1b 1b	2b	6b 6b	2b	30b	8b
CF	CF	CF	CF	CFAP	CF	CF	CF

*FIG. 14.*

CV-ART DAT. COMP SCH.	EXPORT- KONTR.	VERWENDUNG E D MG MV	AV	SOFTWARE CV Verw Version	LÄNGE	RESER- VIERT	PARI- TÄT
4b 3b	1b	1b 1b 1b 1b	2b	6b 6b	2b	28b	8b
CF	CF	CF	CF	CFAP	CF	CF	CF

*FIG. 15.*

CV-ART DAT. XLT SCH.	EXPORT- KONTR.	VERWENDUNG XDin XDout	AV	SOFTWARE CV Verw Version	LÄNGE	RESER- VIERT	PARI- TÄT
4b 3b	1b	1b 1b	2b	6b 6b	2b	30b	8b
CF	CF	CF	CF	CFAP	CF	CF	CF

FIG. 16.

CV-ART DAT. ANSI SCH.	EXPORT- KONTR.	VERWENDUNG E D MG MV ACMD	AV	SOFTWARE CV Verw Version	LÄNGE	RESER- VIERT	PARI- TÄT
4b 3b	1b	1b 1b 1b 1b 1b	2b	6b	2b	27b	8b
CF	CF	CF	CF	CFAP	CF	CF	CF

FIG. 17.

CV-ART PIN PEK SCH.	EXP. KTRL	PB ERST	PIN GEN	PIN PR.	VERWENDUNG PIN PIN XPIN ein aus	AV	SOFTWARE CV Verw Version	LÄNGE	RESER- VIERT	PARI- TÄT
4b 3b	1b	1b	1b	1b	1b	2b	6b 6b	2b	27b	8b
CF	CF	CF				CF	CFAP	CF	CF	CF



FIG. 18.

CV-ART PIN PGK SCH.	EXP.- KTRL.	VERWENDUNG GENPIN GPIN VERPIN VPIN			AV SOFTWARE CV Verw Version	LÄNGE	RESER- VIERT	PARI- TÄT
4b 3b	1b	2b	1b	1b	2b	6b 6b	2b	27b 8b
CF	CF	CF			CF	CFAP	CF	CF

FIG. 19.

CV-ART KEK SEN- DER	EXPORT- KONTR.	VERWENDUNG GKS RFMK XLT SCH aus		SCHL FORM STEUER	AV SOFTWARE CV Verw Version	LÄNGE	RES.	PARI- TÄT
4b 3b	1b	3b	1b	1b	2b	6b 6b	2b	23b 8b
CF	CF	CF		CF	CF	CFAP	CF	CF

FIG. 20.

CV-ART KEK EMPF	EXPORT-VERWENDUNG KONTR.	GKS RPKM XLT SCH aus	SCHL FORM	VERB. STEUER.	AV SOFTWARE CV Verw Version	LÄNGE	RES.	PARI- TÄT
4b 3b	1b	2b 1b 1b	2b	2b	6b 6b	2b	24b	8b
CF	CF	CF	CF	CF	CFAP	CF	CF	CF

FIG. 21.

CV-ART KEK ANSI	EXPORT- KONTR.	VERWENDUNG ARFMK ARTMK AXLT KEY	SCHL FORM	VERB. STEUER. = B'00'	AV SOFTWARE CV Verw Version	LÄNGE	RES.	PARI- TÄT
4b 3b	1b	1b 1b 1b 1b	2b	2b	6b 6b	2b	24b	8b
CF	CF	CF.	CF.	CF	CFAP	CF	CF	CF

*FIG. 22.*

CV-ART SCHL 000 TEIL	EXPORT- KONTR.	SCHL FORM	AV	SOFTWARE CV Verw Version	LÄNGE	RESERV.	PARI- TÄT
4b    3b	1b	2b	2b	6b            6b	2b	30b	8b
CF	CF	CF	CF	CFAP	CF	CF	CF

*FIG. 23.*

CV-ART ICV 000	EXPORT- KONTR.	SOFTWARE CV Verw Version	AV	LÄNGE	RESERV.	PARI- TÄT
4b    3b	1b	6b            6b	2b	2b	32b	8b
CF	CF	CFAP	CF	CF	CF	CF

*FIG. 24.*

CV-ART TOKEN 000	EXPORT- KONTR.	AV	SOFTWARE CV Verw Version	LÄNGE	RESERV.	PARI- TÄT
4b    3b	1b	2b	6b            6b	2b	32b	8b
CF	CF	CF	CFAP	CF	CF	CF

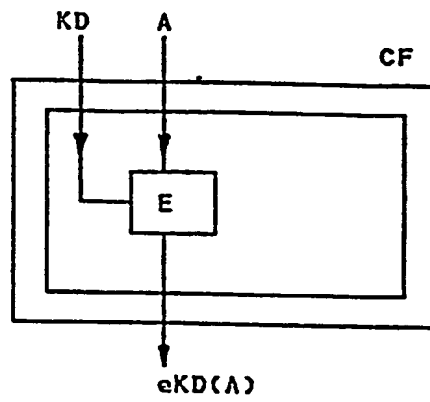
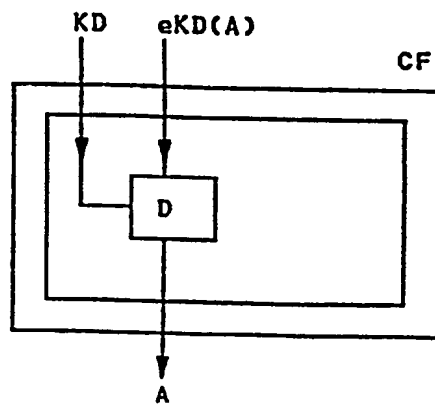
*FIG. 25.**FIG. 26.*

FIG. 27

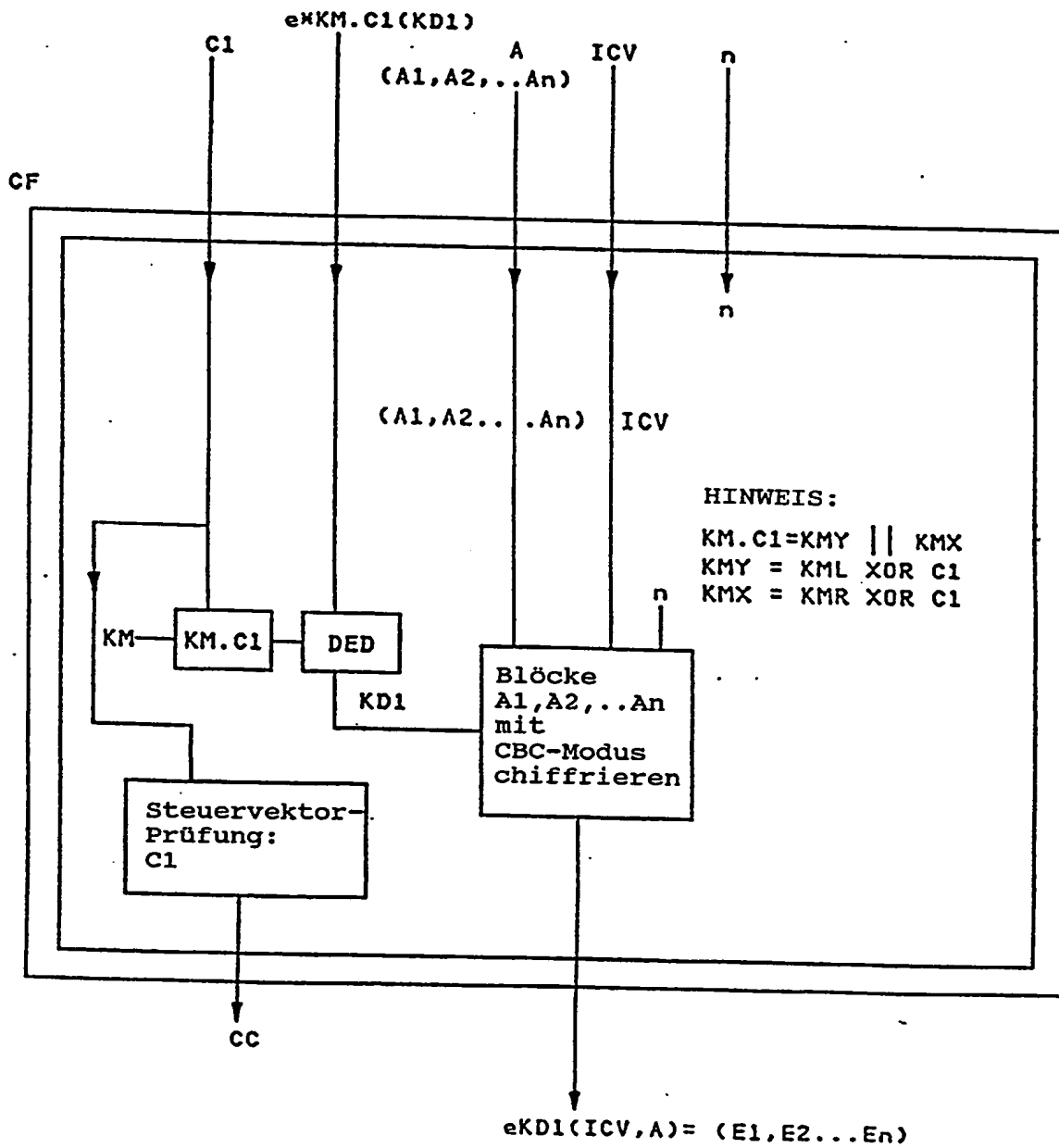


FIG. 28.

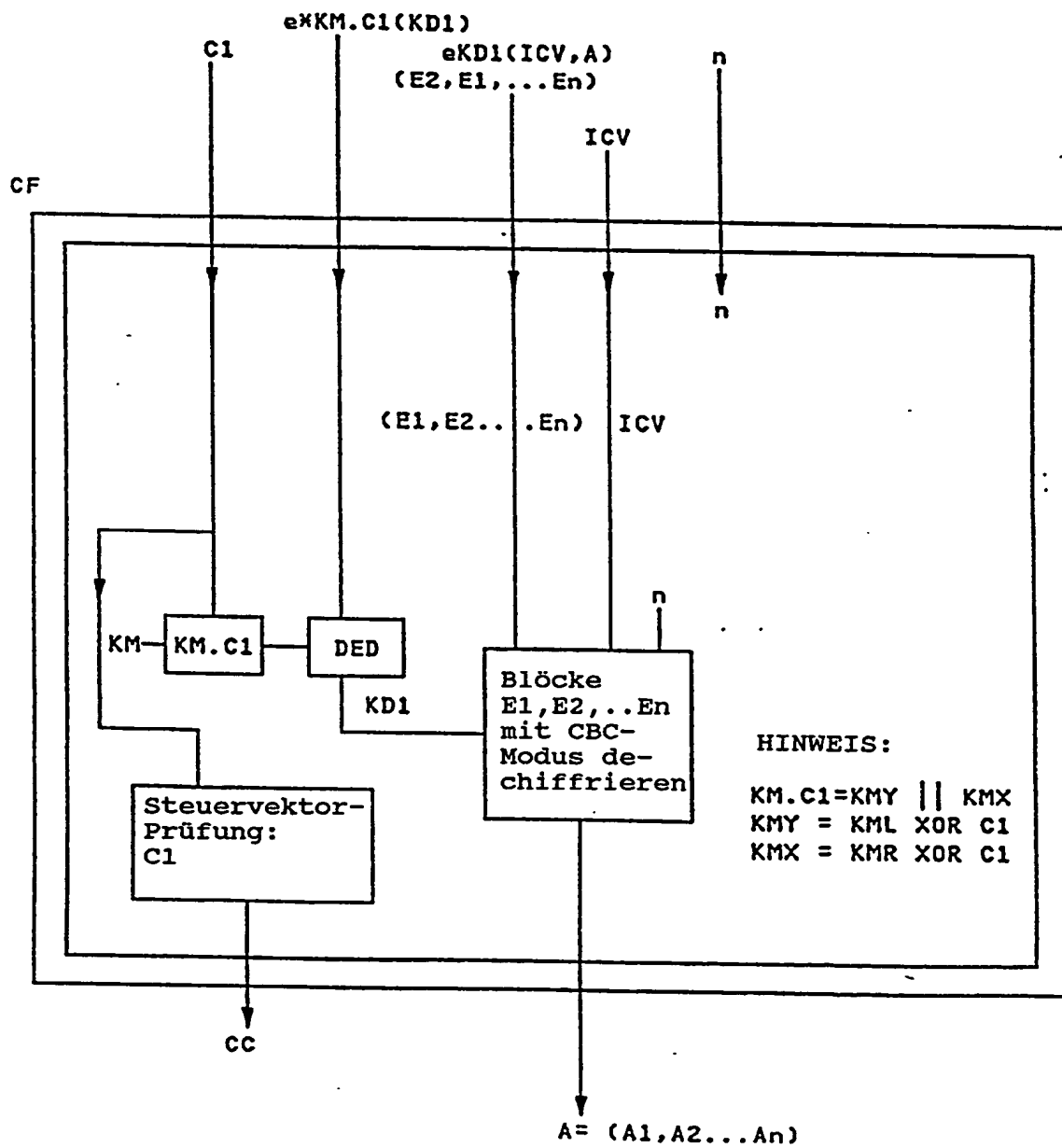
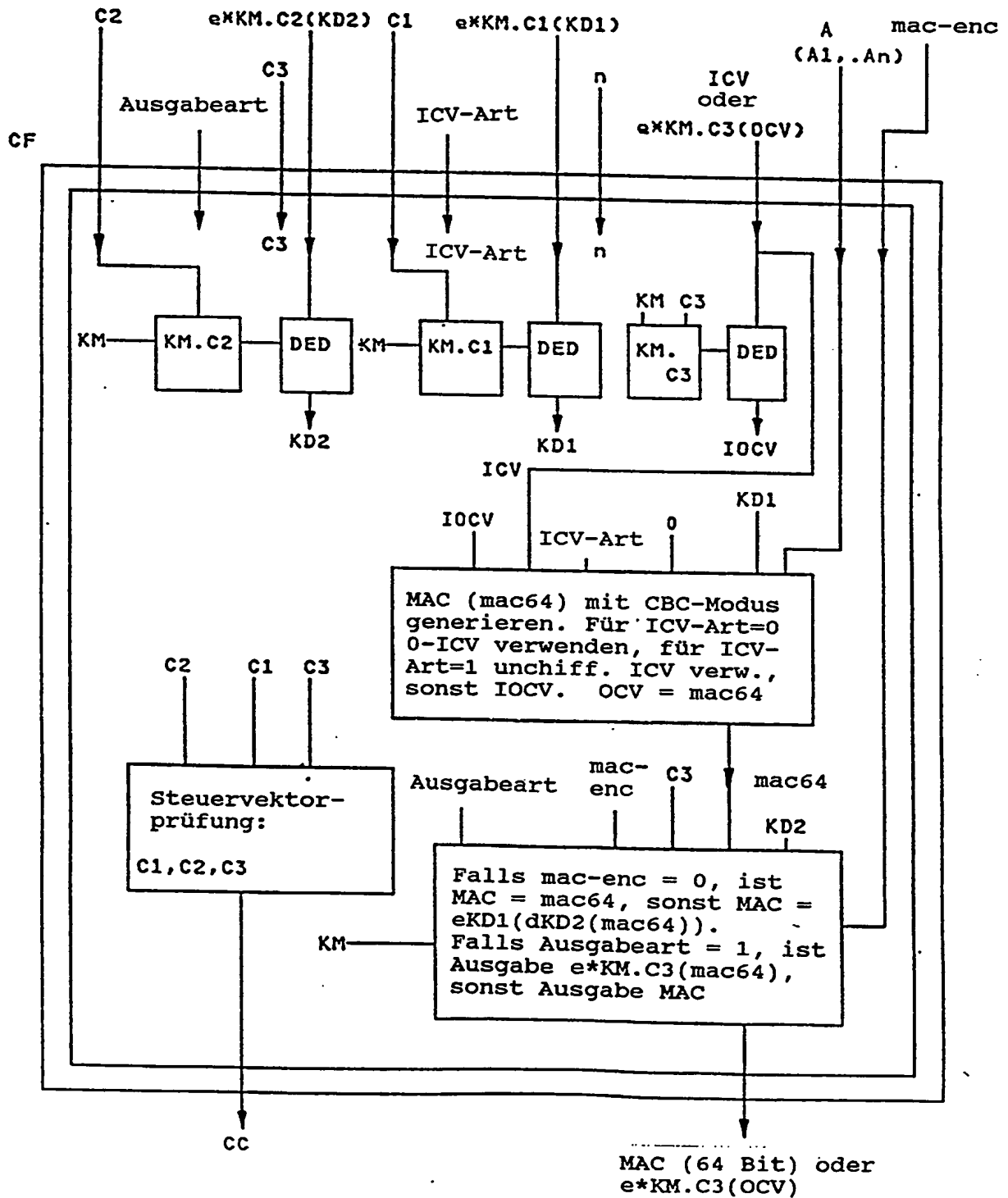


FIG. 29.



19/58

FIG. 30.

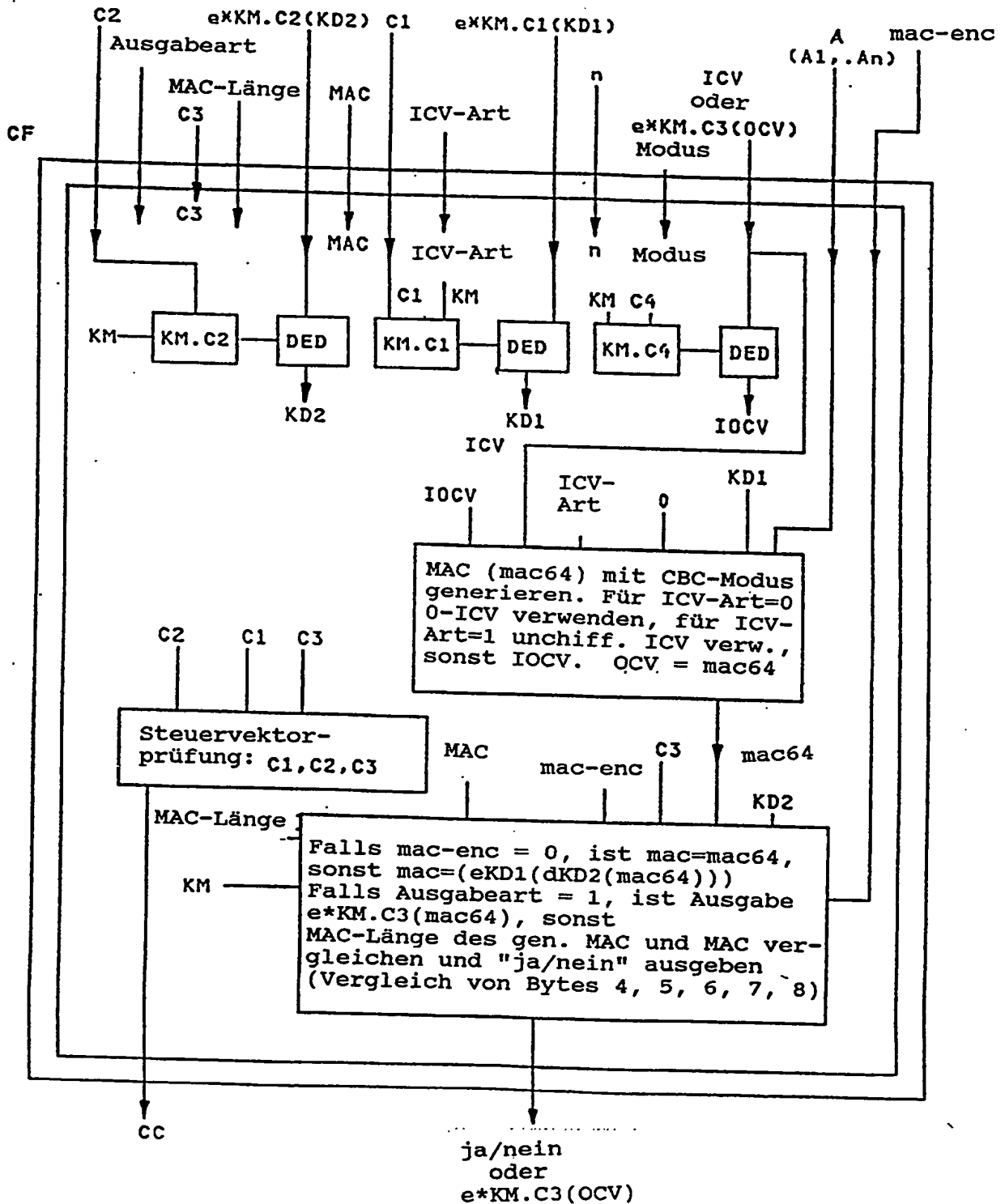
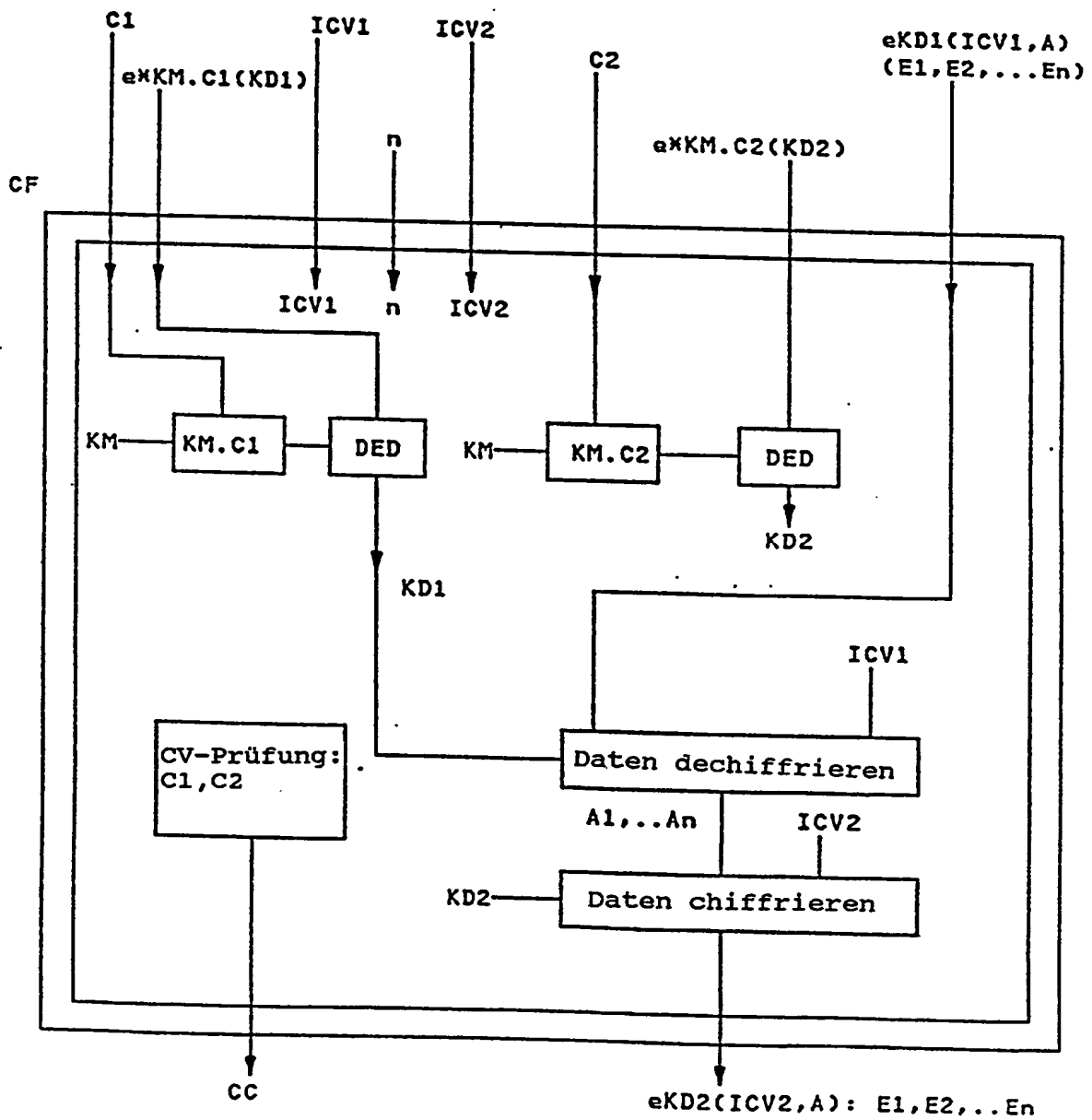




FIG. 31.



*FIG. 32.*

OP-OP	
OP-IM	
OP-EX	
EX-EX	
IM-EX	

*FIG. 34.*

	C3	C4
CV-Art	Daten/Vertr.	Daten/Vertr.
	Daten/MAC	Daten/MAC

*FIG. 35.*

	C2L	C2R
Schlüssel- sel- form	00	00
	01	01
	10	11
	11	10
Verb. steuer.	01 (nur CV)	01 (nur CV)

**FIG. 33.**

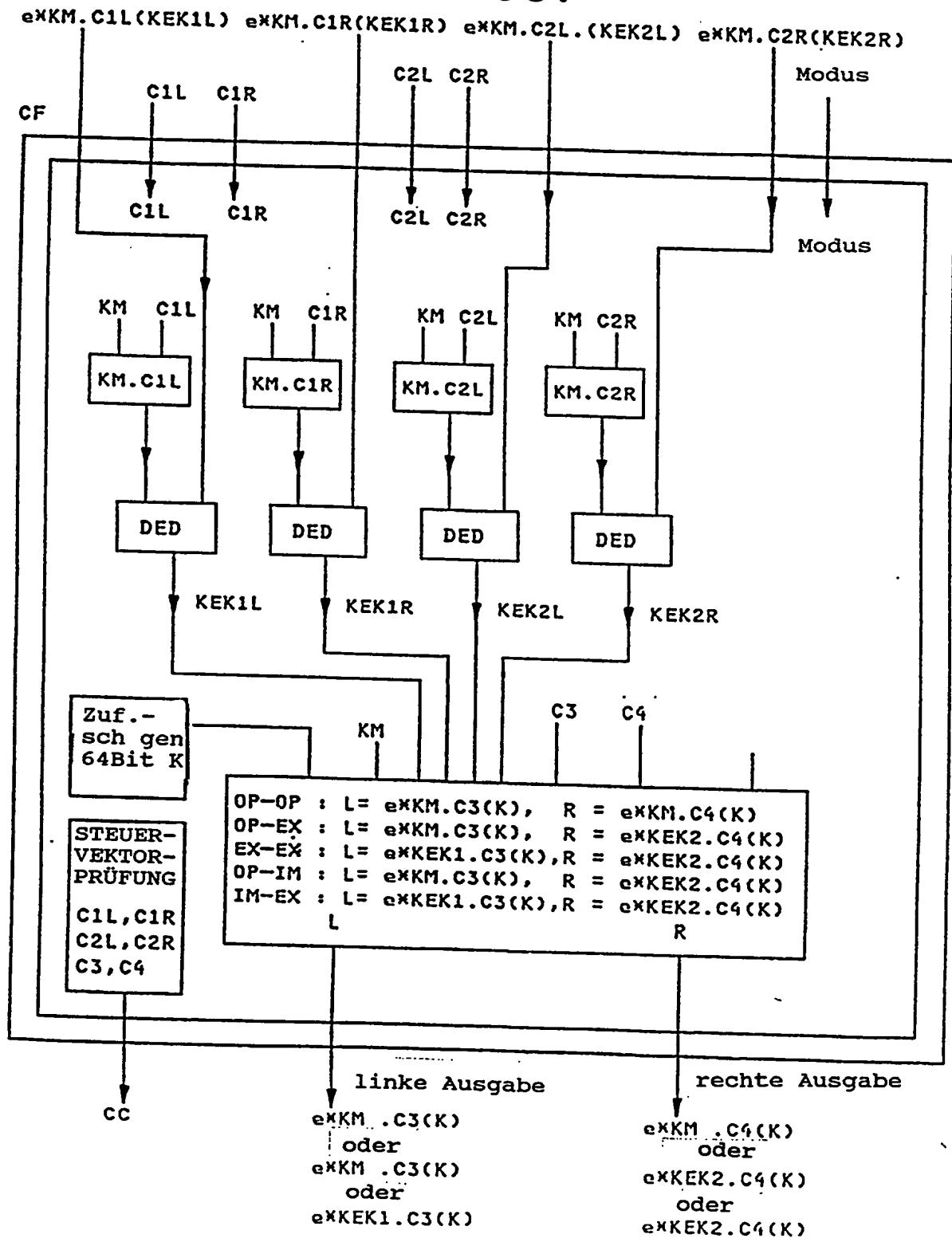


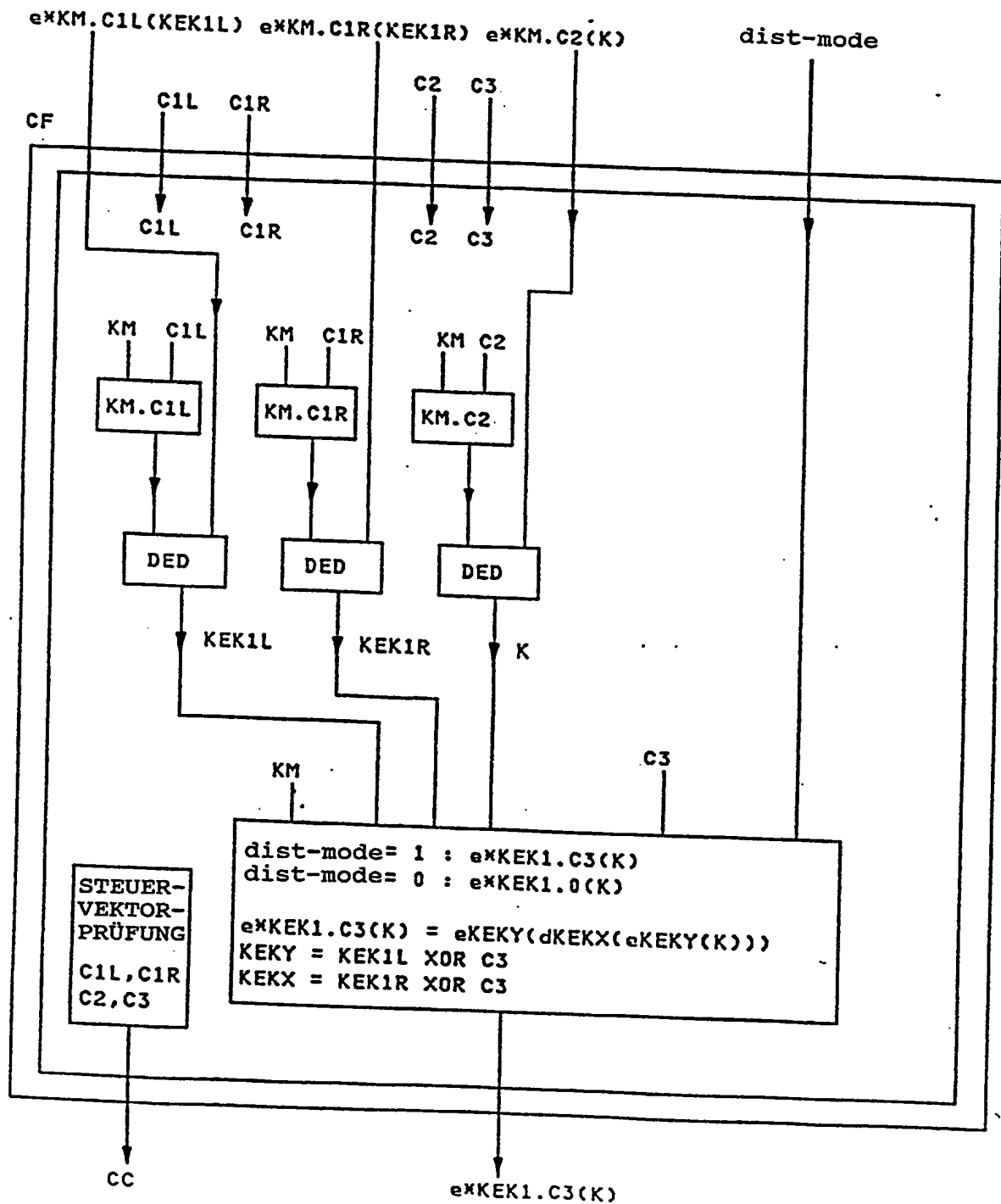
FIG. 36.

	C3	C4
CV-Art	Daten/Vertr.	Daten/Vertr.
	Daten/Vertr.	Daten/XLT
	Daten/XLT	Daten/Vertr.
	Daten/XLT	Daten/XLT
	Daten/MAC	Daten/MAC
	PIN/PEK	PIN/PEK
	KEK/Sender	KEK/Empf.
	KEK/Empf.	KEK/Sender
Verb.- Steuer. bei CV-Art = KEK	01	01
	10	10
	11	11
	00	00
Schl.- Form bei CV-Art = KEK	00	00
	01	01
	10	10
	11	11
	01	00
	10	00
	11	00

FIG. 37.

	C3	C4
CV-Art	Daten/Vertr.	Daten/Vertr.
	Daten/Vertr. & E = 1	Daten/XLT & XDin = 1
	Daten/XLT & XDin=1	Daten/Vertr. & E = 1
	Daten/MAC	Daten/MAC

FIG. 38.



25/58  
**FIG. 39.**

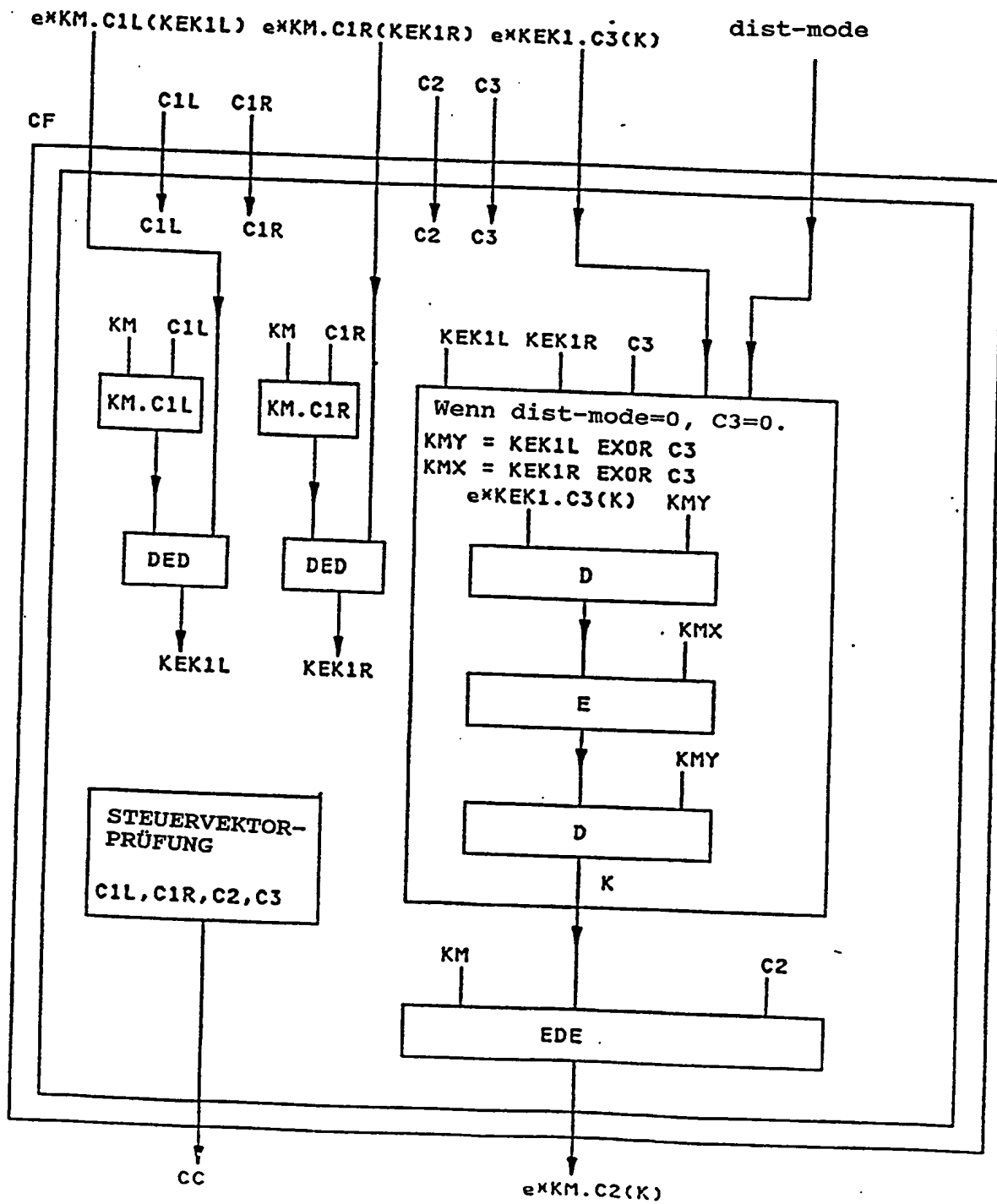


FIG. 40.

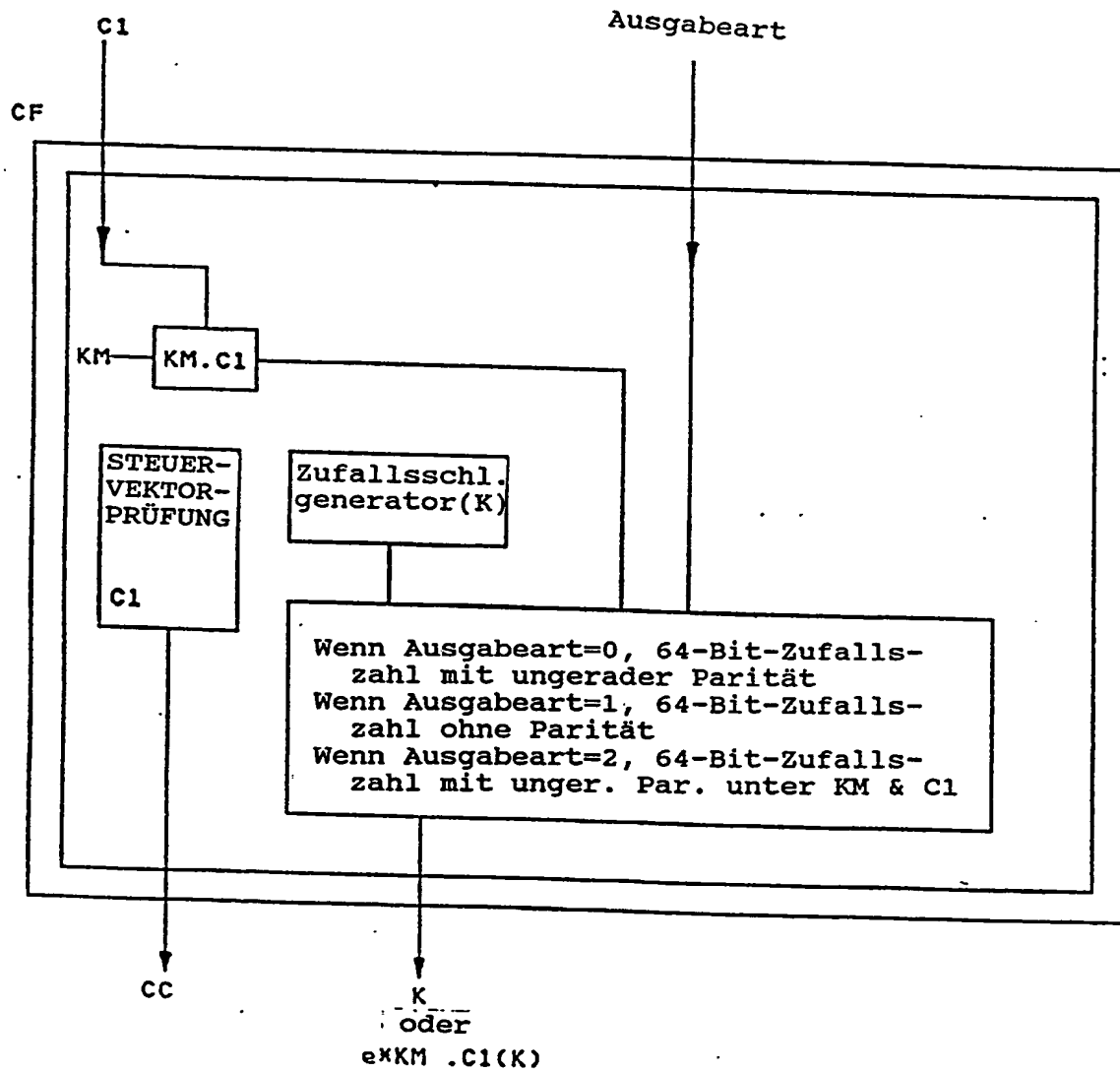


FIG. 41.

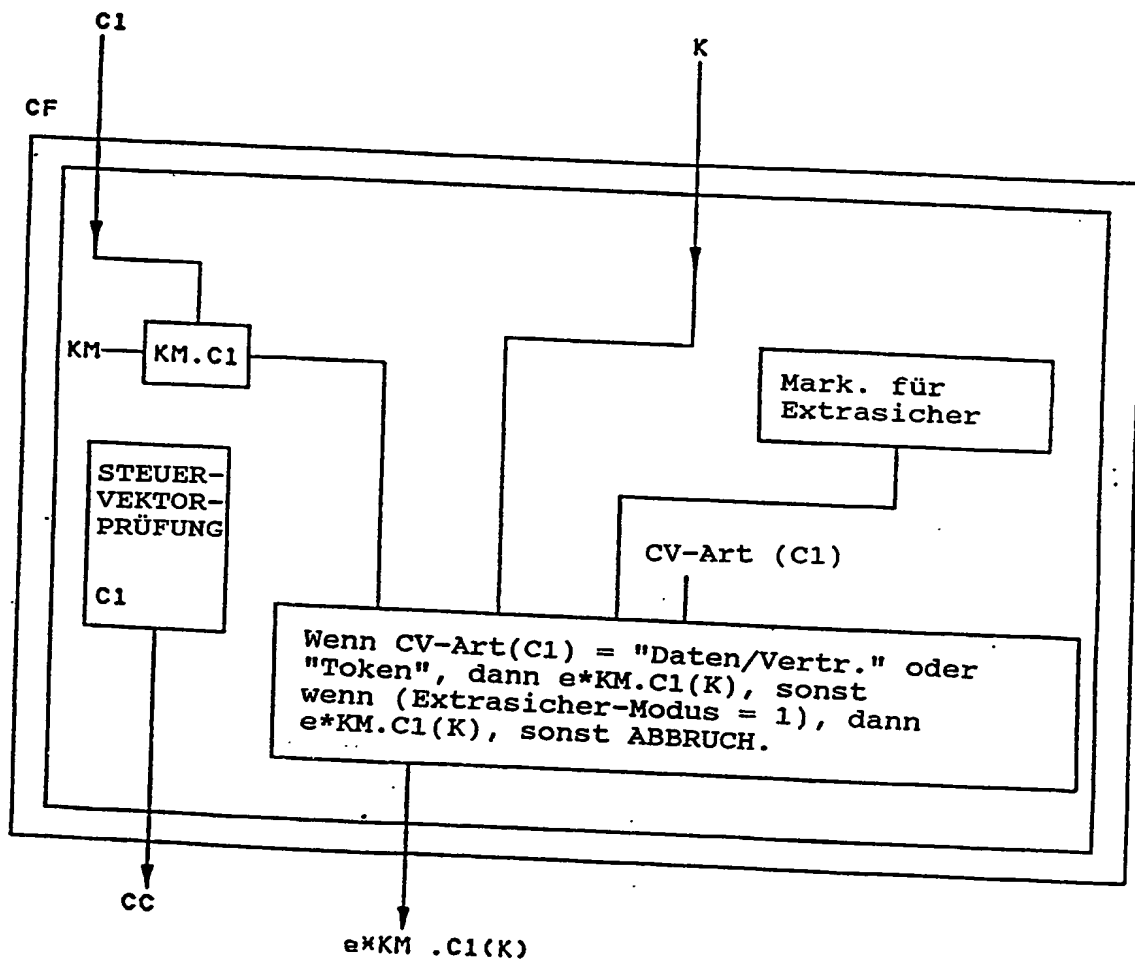




FIG. 42.

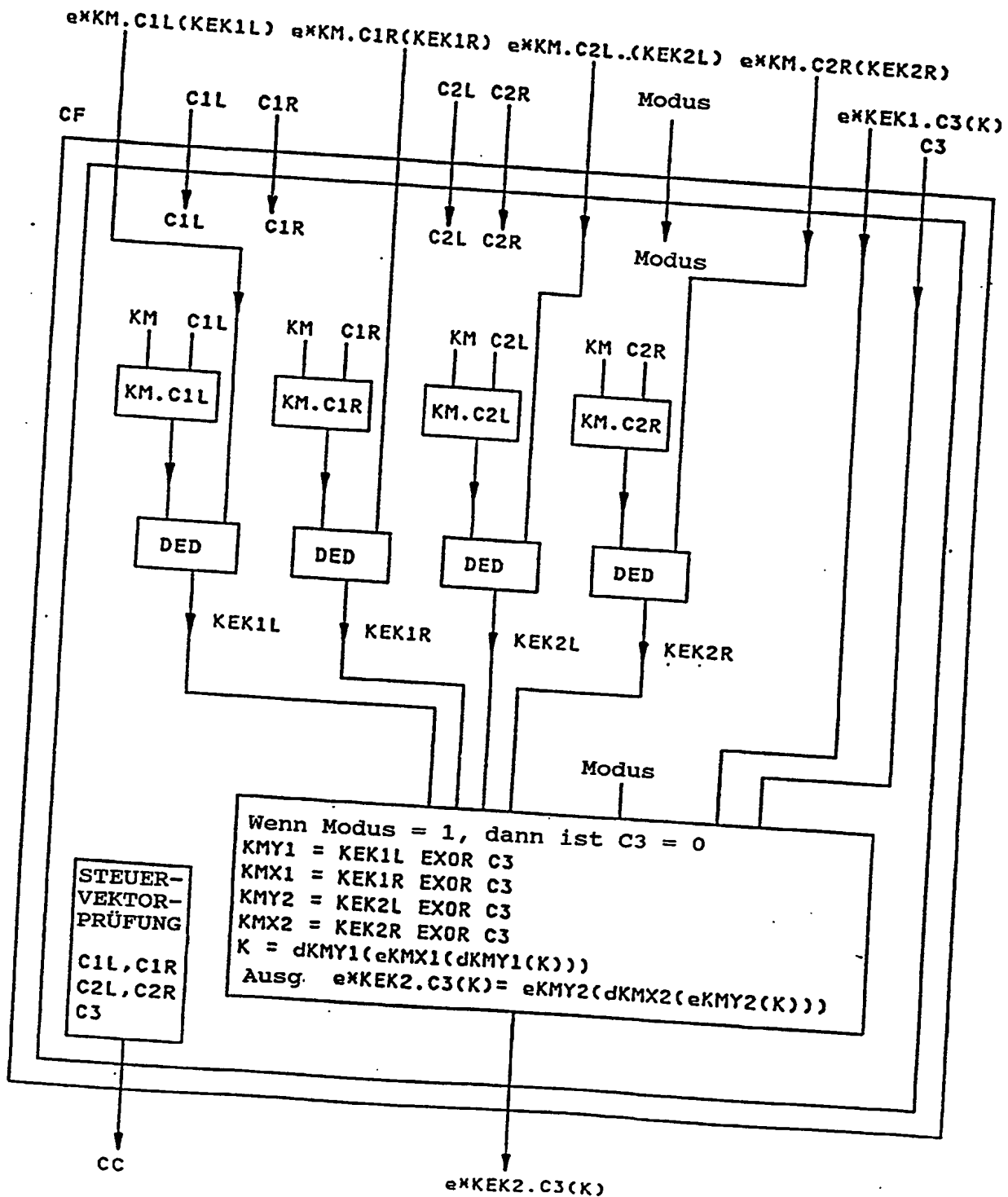
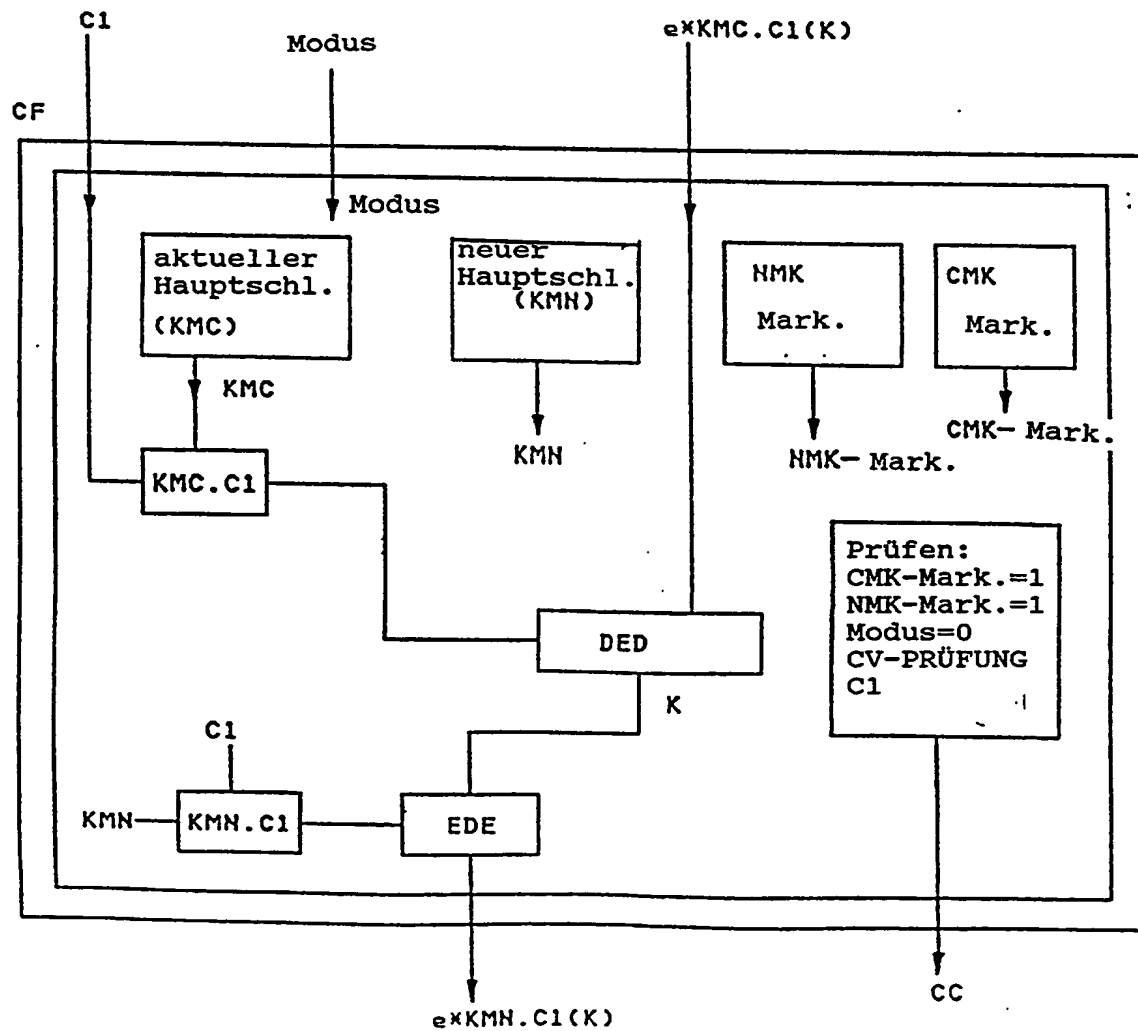


FIG. 43.

	C1L	C1R
Schlüssel- sel- form	00	00
	01	01
	10	11
	11	10

FIG. 44.



30/58

FIG. 45.

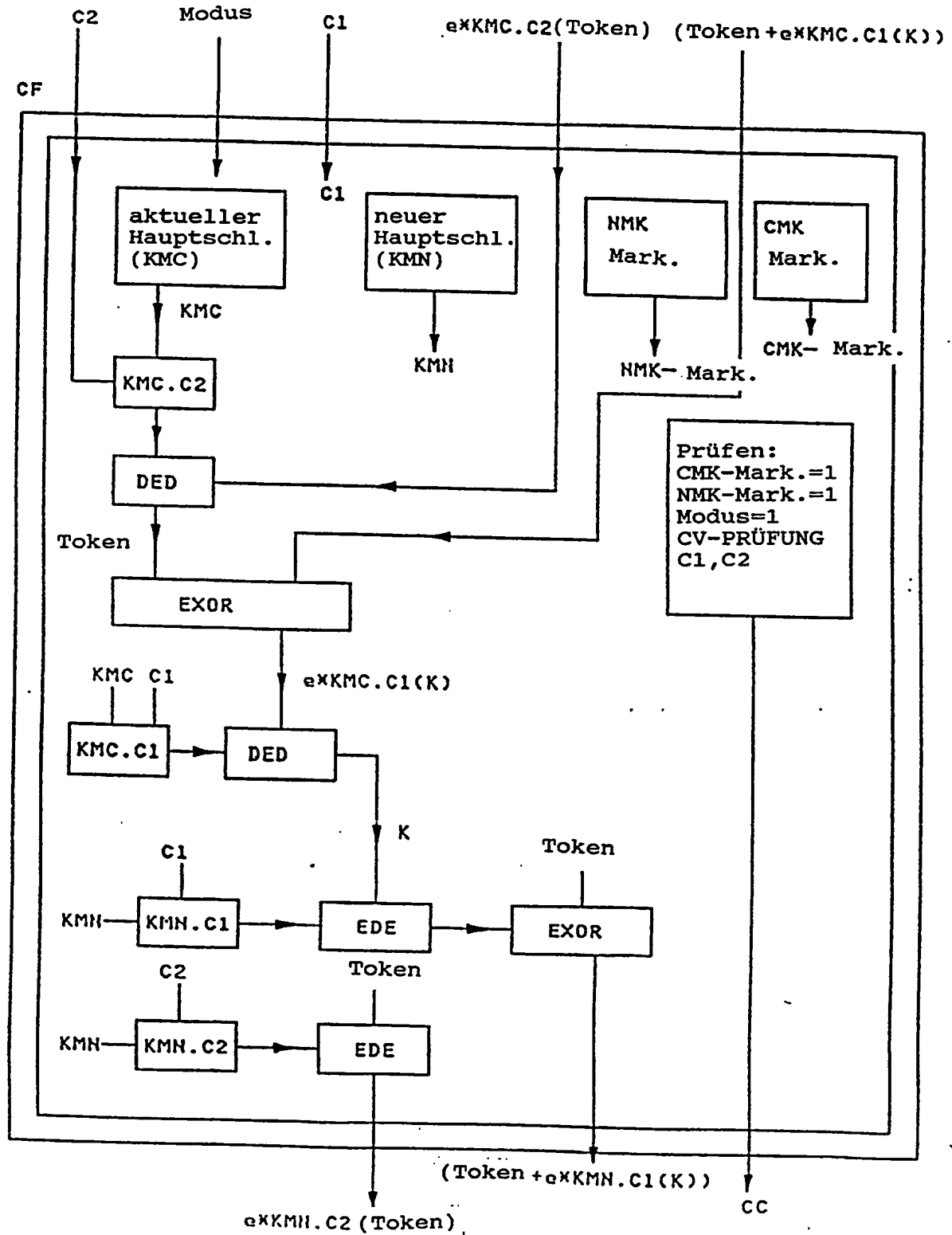
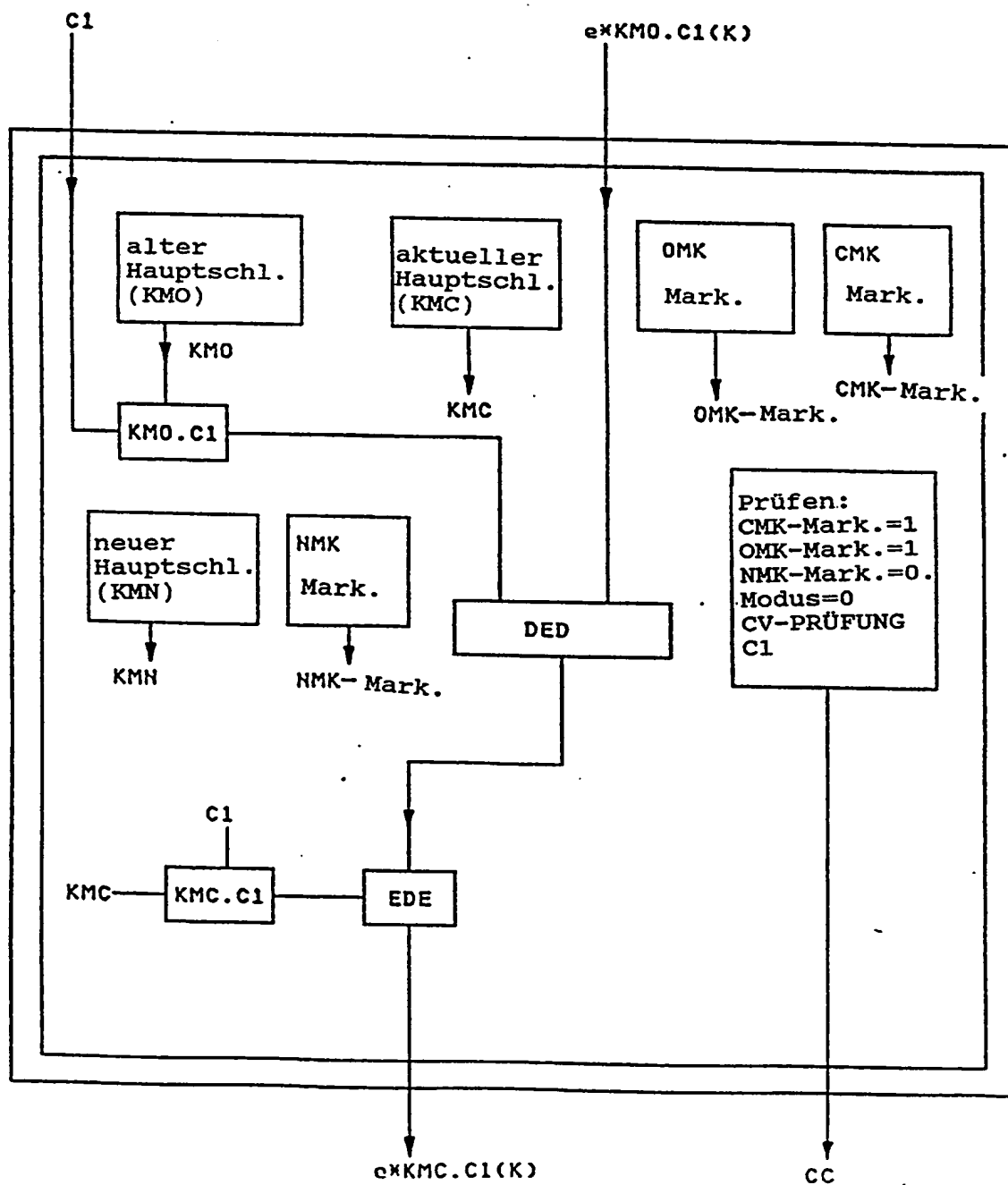


FIG. 46.



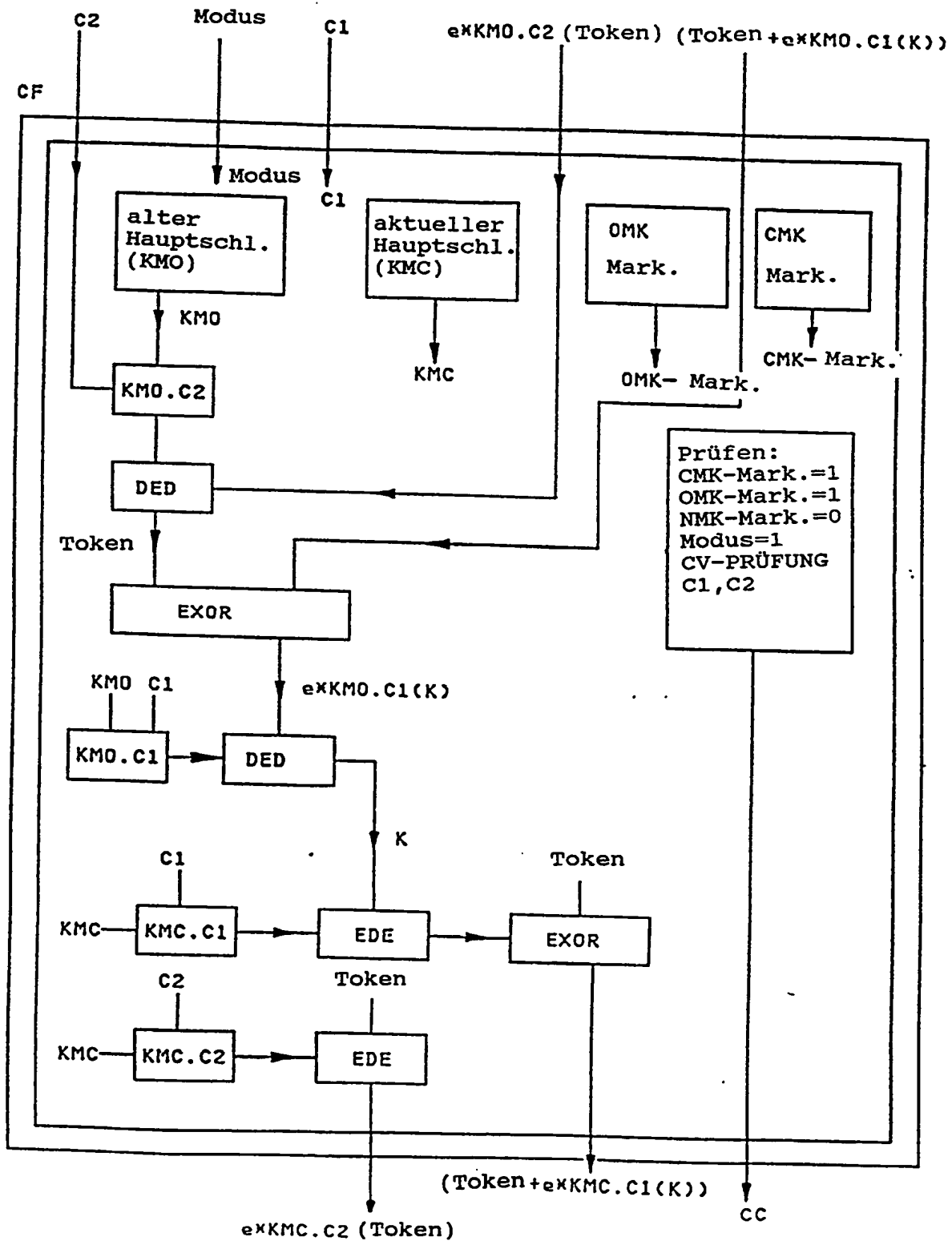
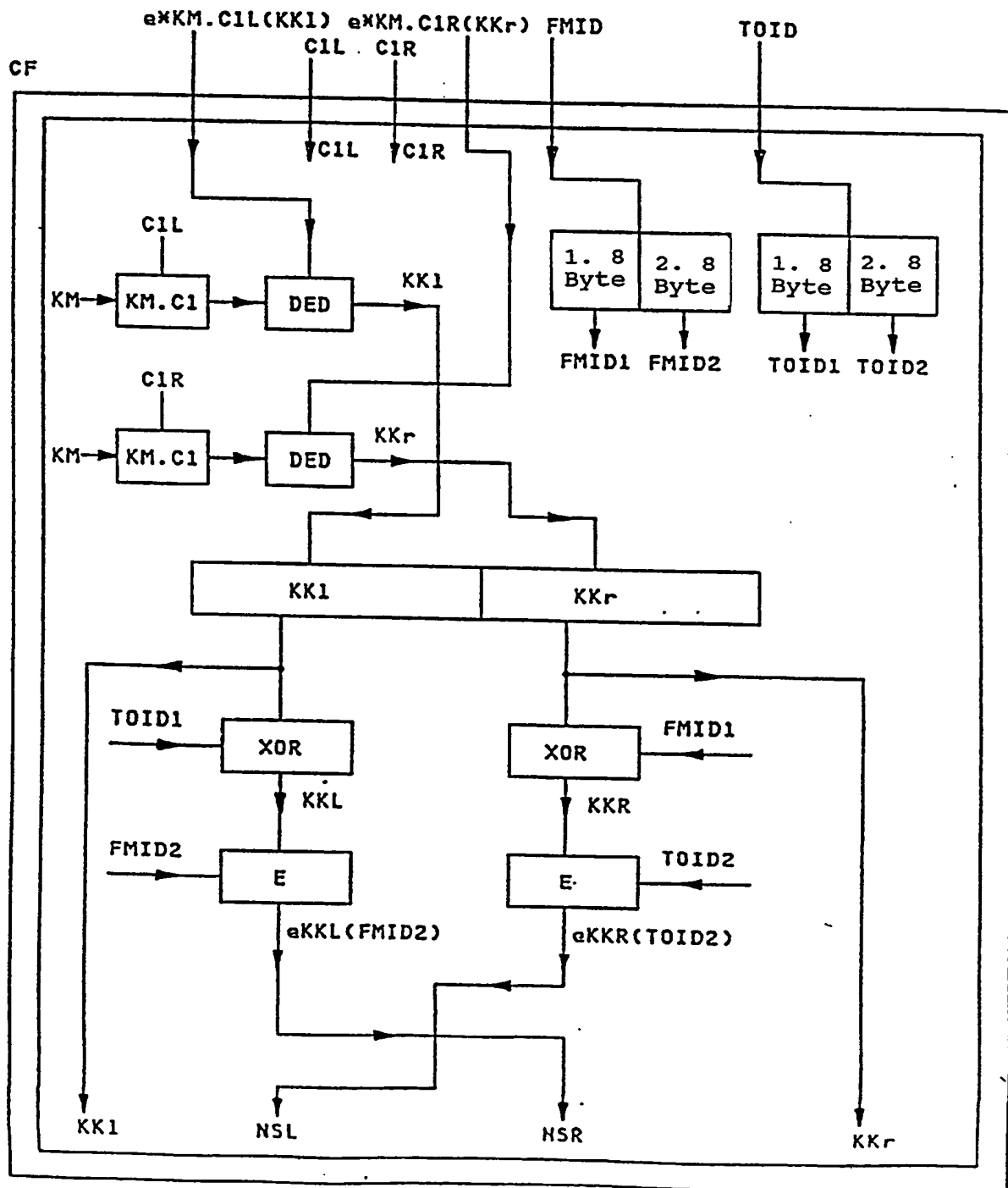


FIG. 48.



34/58  
**FIG. 49.**

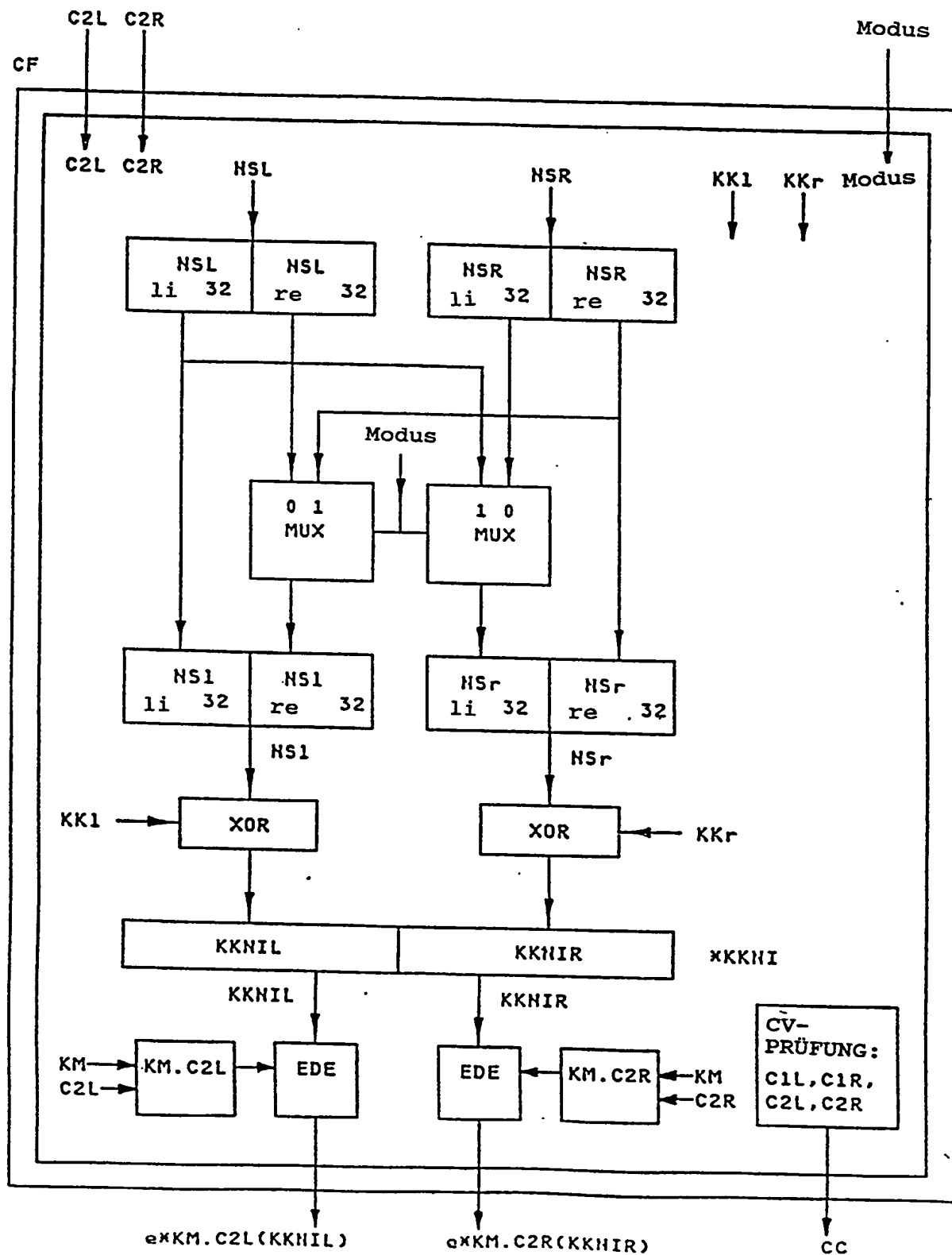
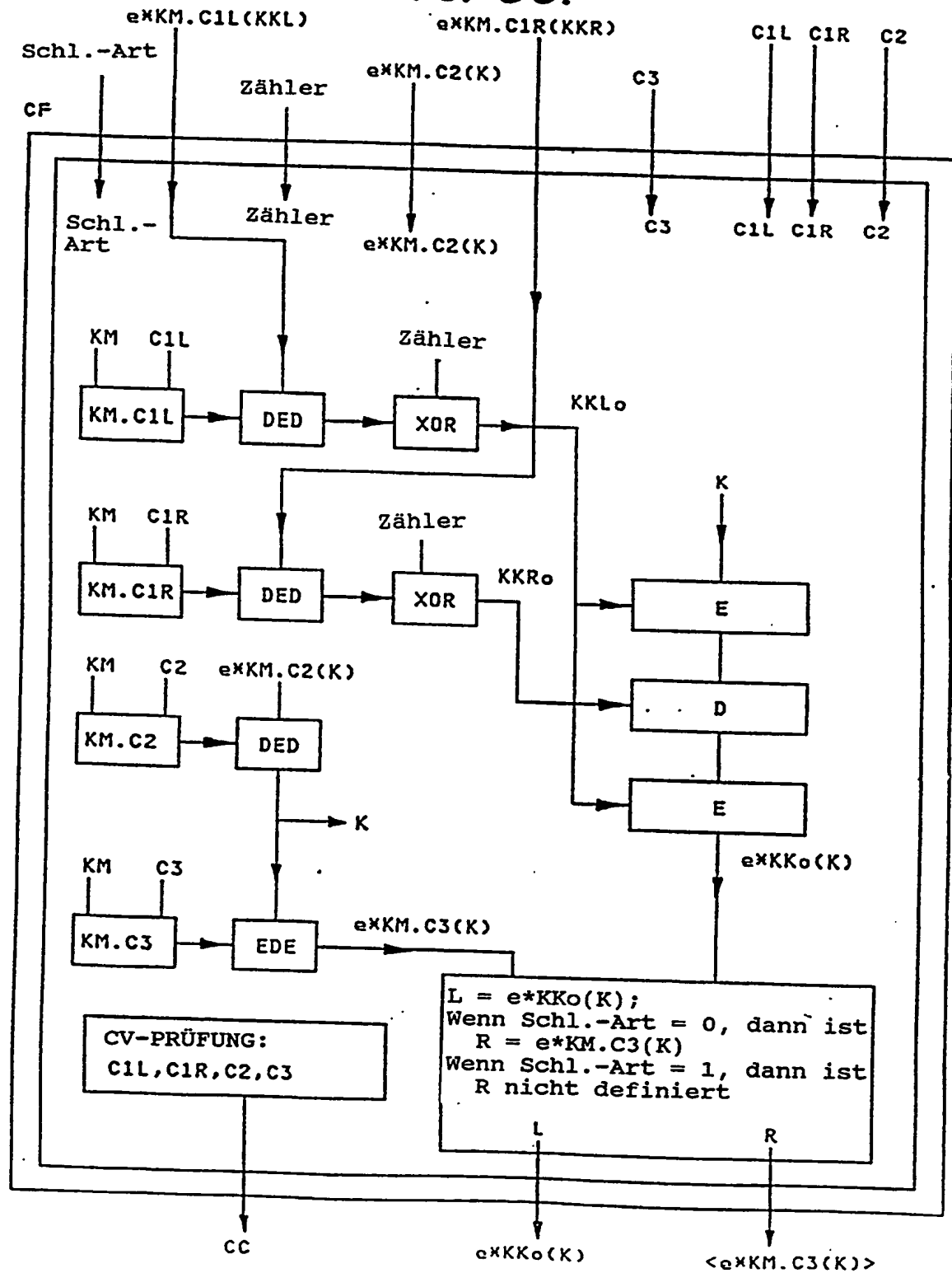


FIG. 50.





*FIG. 51.*

C2

E	D	MG	MV	ACMB
1	1	0	0	0
0	0	1	1	0

*FIG. 53.*

C3

E	D	MG	MV	ACMB
0	0	1	1	0
0	0	0	0	1

*FIG. 55.*

C2

E	D	MG	MV	ACMB
1	1	0	0	0
0	0	1	1	0

*FIG. 57.*

C3

E	D	MG	MV	ACMB
0	0	1	1	0
0	0	0	0	1

*FIG. 52.*

C2 & C1L, C1R

CV-Art von C2	Schlüsselform		
	C2	C1L	C1R
Daten/ANSI	N/A	01	01
		10	11
KEK/ANSI	01	01	01
		10	11
	10	10	11
	11	10	11

*FIG. 56.*

C2 & C1L, C1R

CV-Art von C2	Schlüsselform		
	C2	C1L	C1R
Daten/ANSI	N/A	01	01
		10	11
KEK/ANSI	01	01	01
		10	11
	10	10	11
	11	10	11

FIG. 54.

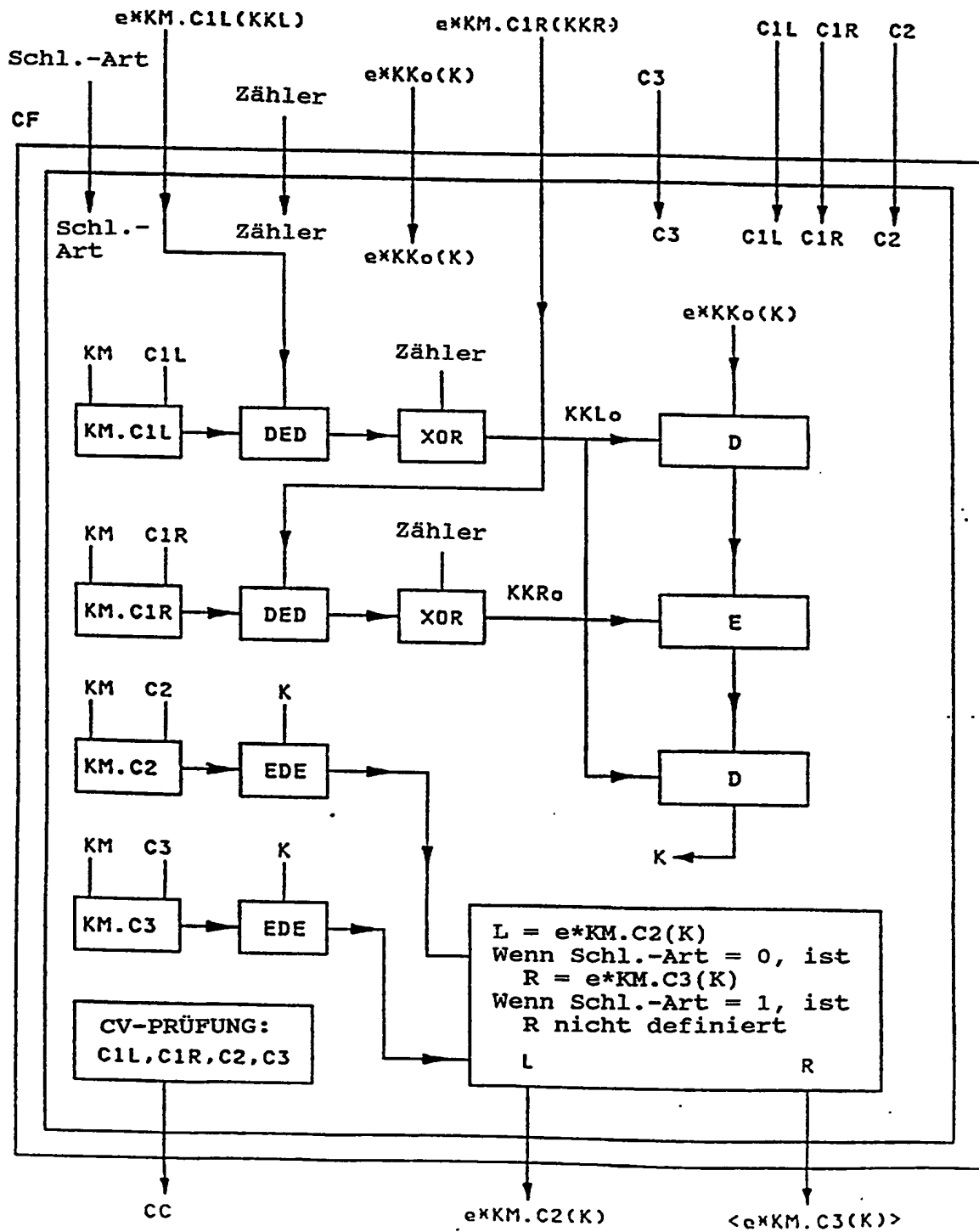


FIG. 58.

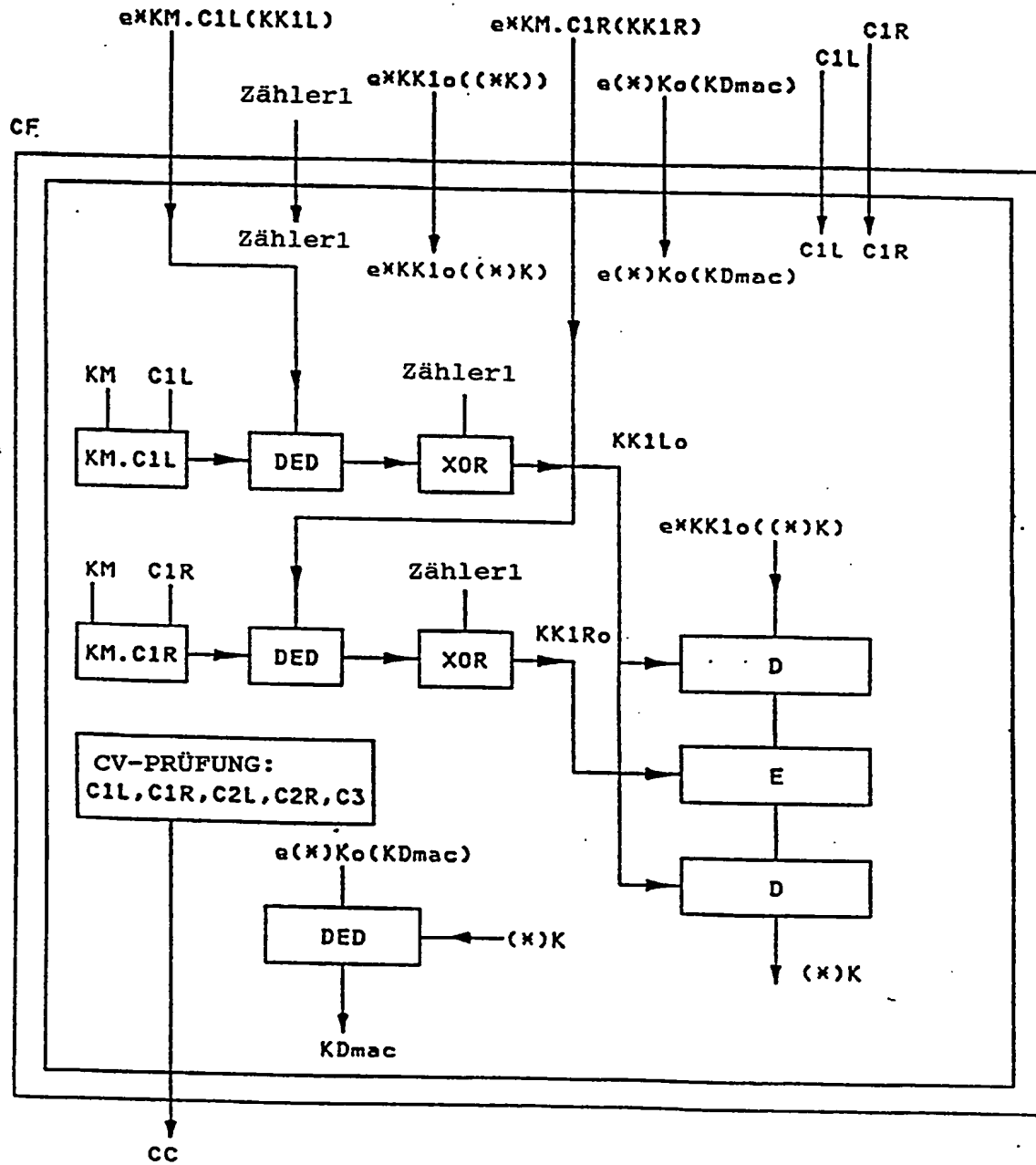
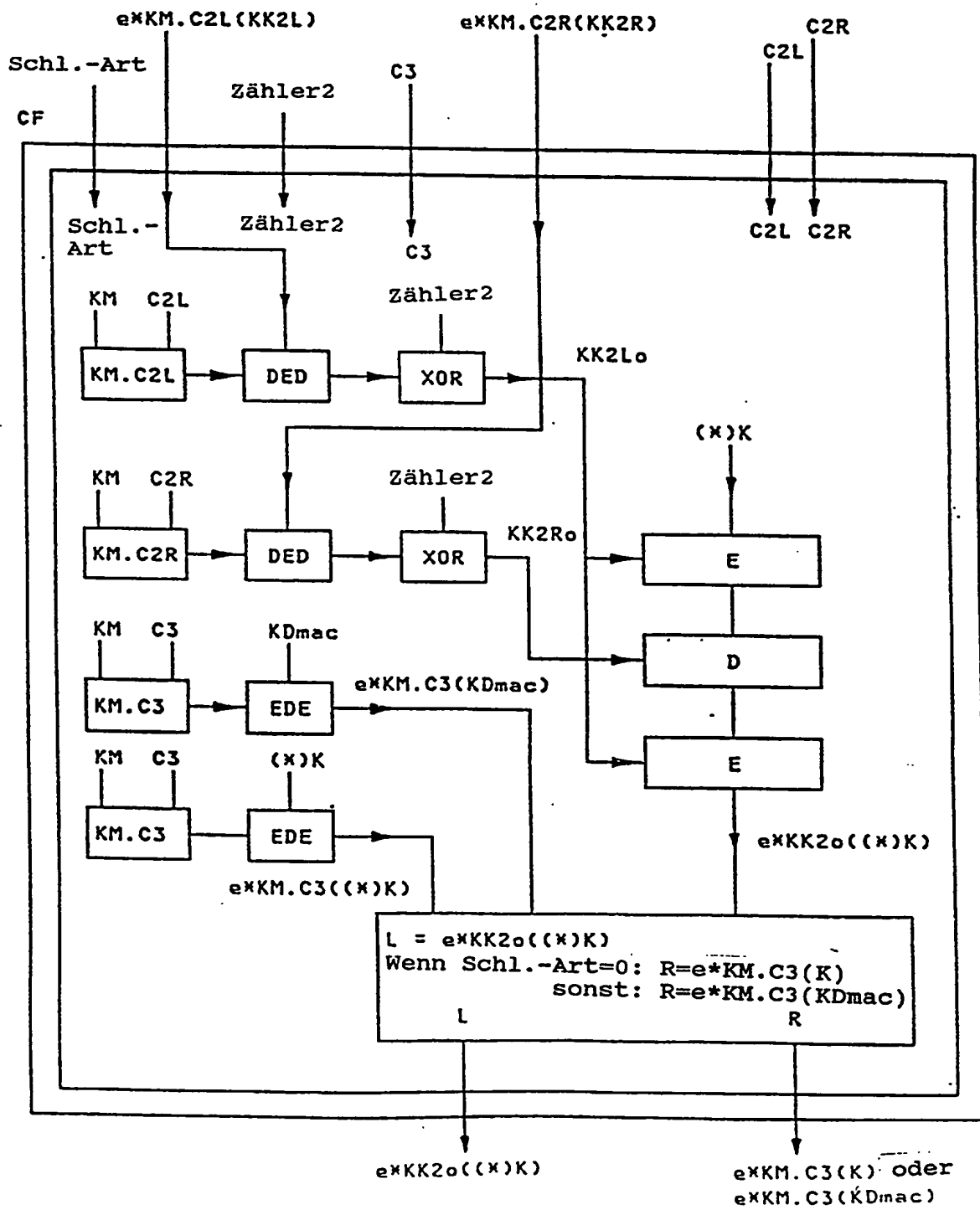
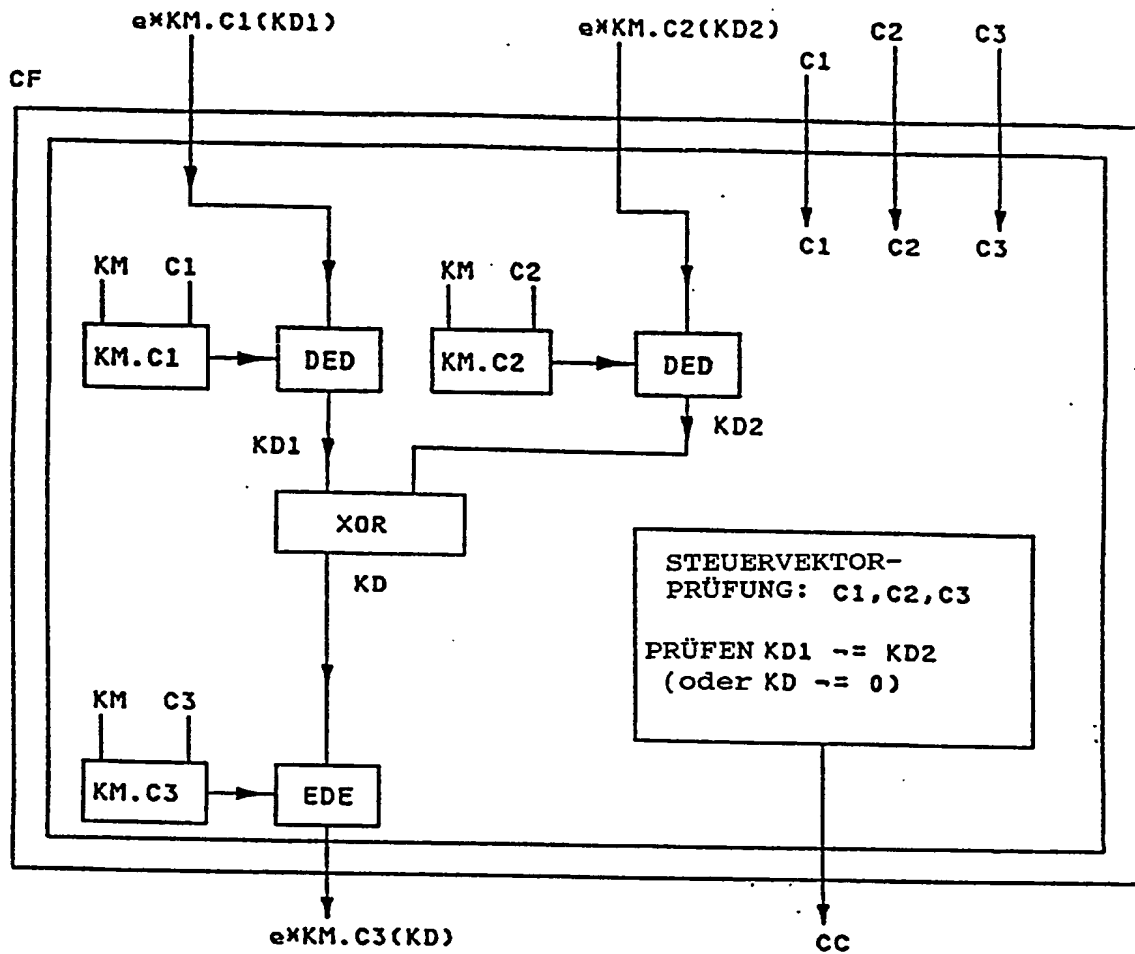


FIG. 59.



40/58  
**FIG. 61.**



**FIG. 60.**

C3

E	D	MG	MV	CMBKD
0	0	0	0	1
0	0	1	1	0

**FIG. 62.**

C1

E	D	MG	MV	ACMB
0	0	0	0	1

**FIG. 63.**

C2

E	D	MG	MV	ACMB
0	0	0	0	1

**FIG. 64.**

C3

E	D	MG	MV	ACMB
0	0	1	1	0

*FIG. 65.*

Schlüsselverwaltung-Aufgabe		CA-Anweisungen
Haupt-schlüssel initial.	<ul style="list-style-type: none"> <li>- Manuelle Eingabe des HS</li> <li>- Aktivierung des HS</li> </ul>	LFMK, CMKP, CVP (nicht obligatorisch, s. Hinweis 1), SMK
KEK initial.	<ul style="list-style-type: none"> <li>- Manuelle Eingabe von KEKs über die phys. Schnittstelle</li> <li>- Eingabe von KEKs über die Programmschnittstelle</li> </ul>	LFKP, KCP, CVP (nicht obligatorisch, s. Hinweis 1) EMK
Unchiff. oder (un)chiff. Schlüssel generieren	<ul style="list-style-type: none"> <li>- Generieren unchiff. Schl. über Programmschnittstelle</li> <li>- Generieren unchiff. Schl. zur Versendung durch Kurier und für lokale Speicherung in chiffrierter Form</li> </ul>	KGEN KGEN, EMK
Schlüssel für lokale Verwendung generieren	<ul style="list-style-type: none"> <li>- Generieren eines Schl. für lok. Verw. und Speicherung</li> <li>- Generieren zweier Schlüssel-exemplare für lokale Verwendung, mit verschiedenen Attributen; Schlüsselart:               <ul style="list-style-type: none"> <li>a. Daten/Vertraulichkeit</li> <li>b. Daten/MAC</li> <li>c. Daten/andere Unterarten</li> <li>d. KEK</li> <li>e. PIN-Schlüssel</li> </ul> </li> </ul>	KGEN GKS (op-op-Modus) GKS (op-op-Modus) Von CA untersagt Von CA untersagt Von CA untersagt

Hinweise: 1. Die Anweisungen LFMKP, CMKP, LKP, CKP und CVP sind nicht obligatorisch. Jedes System kann seine eigene systemspezifische Schlüssellade-Anweisung besitzen.

FIG. 66.

	Schlüsselverwaltungs- Aufgabe	CA-Anweisungen
Schlüssel für lokale Verwendung und Export generieren	<ul style="list-style-type: none"> <li>- Zwei Schlüsselexemplare mit versch. Verwendungsattributen erstellen, eins für lok. Verw. und eins für Export über CV-Kanal. Schlüsselart: a. Daten/Kompatibilität b. Daten/andere Unterarten c. KEK d. PIN/PEK</li> </ul>	<p>Nicht von CA unterst. GKS (op-ex-Modus) GKS (op-ex-Modus) GKS (op-ex-Modus)</p>
	<ul style="list-style-type: none"> <li>- Zwei Schlüsselexemplare mit versch. Verwendungsattributen erstellen, eins für lok. Verw. und eins für Export über CV=0-Kanal.</li> </ul>	<p>Nicht von CA unterst.</p>
	<ul style="list-style-type: none"> <li>- Ein Schlüsselexemplar für lok. Verwendung und Export über CV-Kanal generieren.</li> </ul>	<p>KGEN und RFMK</p>
	<ul style="list-style-type: none"> <li>- Ein Schlüsselexemplar für lok. Verwendung und Export über CV=0-Kanal generieren. Schlüsselart: a. Daten/Kompatibilität b. Daten/andere Unterarten c. KEK d. PIN-Schlüssel</li> </ul>	<p>KGEN und RFMK Von CA untersagt Von CA untersagt Von CA untersagt</p>

FIG. 67.

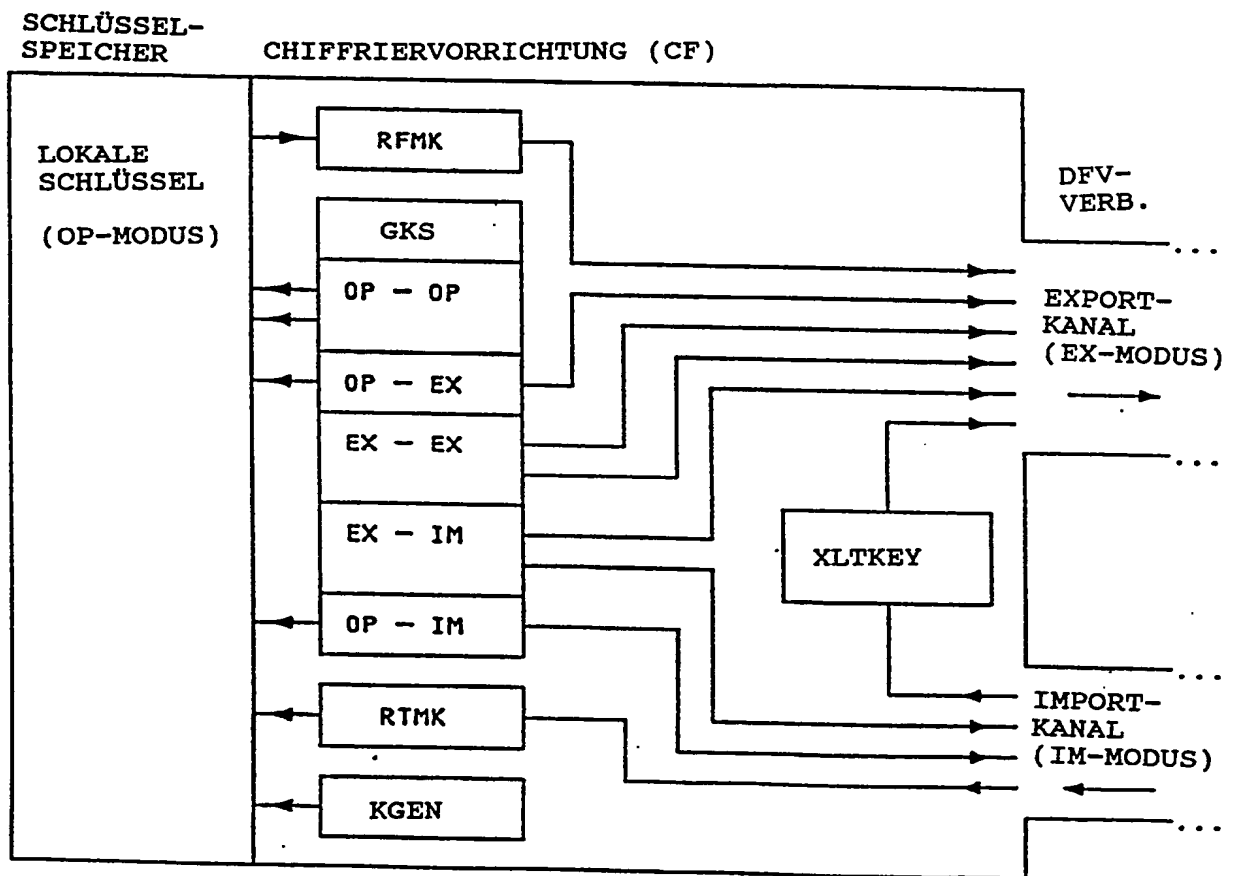
Schlüsselverwaltungs- Aufgabe	CA-Anweisungen
Schlüssel für lokale Verwendung und Export an n Knoten generieren ( $n > 1$ )	<ul style="list-style-type: none"> <li>- Zwei Schlüsselexemplare mit versch. Verwendungsattributen generieren, eins für lok. Verw. und eins für Export an n Knoten über CV-Kanal.</li> </ul>
	<ul style="list-style-type: none"> <li>- Zwei Schlüsselexemplare mit versch. Verwendungsattributen generieren, eins für lok. Verw. und eins für Export an n Knoten über CV=0-Kanal.</li> </ul>
	<ul style="list-style-type: none"> <li>- Ein Schlüsselexemplar für lok. Verwendung und Export an n Knoten über CV=0-Kanal generieren. Schlüsselart:               <ul style="list-style-type: none"> <li>a. Daten/Kompatibilität KGEN und RFMK</li> <li>b. Daten/andere Unterarten Von CA untersagt</li> <li>c. KEK Von CA untersagt</li> <li>d. PIN-Schlüssel Von CA untersagt</li> </ul> </li> </ul>
	<ul style="list-style-type: none"> <li>- Ein Schlüsselexemplar für lok. Verwendung und Export an n Knoten über CV-Kanal generieren. Schlüsselart beliebig.</li> </ul>



FIG. 68.

Schlüsselverwaltungs- Aufgabe		CA-Anweisungen
Mit CUSP/ 3848 kom- patible Sitzungs- oder Datei- schlüssel gener.	- Mit IBM 3848/CUSP und PCF kompatiblen Sitzungs- oder Dateischlüssel generieren	GKS (op-ex-Modus)
	- Sitzungsschlüssel, SNA- Mehrdomänen, generieren	GKS (im-ex-Modus)
Schlüssel- vertei- lungs- zentrum	- 2 Schlüsselexemplare ohne lok. Verw., mit versch. Verwendungsattributen gene- rieren, eins für elektron. Verteilung über CV-Kanal an 1. Knoten, das andere für elektron. Verteilung über CV-Kanal an einen 2. Kno- ten. Schlüsselart: a. Daten/Kompatibilität b. Daten/andere Unterarten c. KEK d. PIN-Schlüssel	nicht von CA unterst. GKS (ex-ex-Modus) GKS (ex-ex-Modus) GKS (ex-ex-Modus)
	- 2 Schlüsselexemplare ohne lok. Verw., mit versch. Verwendungsattributen gene- rieren, eins für elektron. Verteilung über CV=0-Kanal an 1. Knoten, das andere für elektron. Verteilung über CV=0-Kanal an einen 2. Knoten.	Von CA untersagt
Schlüssel- umwand- lungs- zentrum	- Schlüssel empfangen und um- wandeln	XLTKEY
Import von Schlüsseln	- Elektron. Import über Pro- grammschnittstelle von a. Daten/Kompatibilität über einen CV=0-Kanal	RTMK
	b. beliebige Schlüsselart, CV-Kanal c. alle außer Daten/Komp., über CV=0-Kanal	RTMK Von CA untersagt

FIG. 69.



**Hinweis:** EX-Modus entspricht Chiffrierung unter einem KEK/-Sender.  
 IM-Modus entspricht Chiffrierung unter einem KEK/-Empfänger.  
 OP-Modus entspricht Chiffrierung unter Hauptschlüssel KM.

FIG. 70.

CV-Art/Unterart	Kanalart		
	CV	CV = 0	ANSI
Daten/Vertraulichkeit	j	n	n
Daten/MAC	j	n	n
Daten/XLT	j	n	n
Daten/Kompatibilität	j*	j	n
Daten/ANSI	n	n	j
KEK/Sender	j	n	n
KEK/Empfänger	j	n	n
KEK/ANSI	n	n	j
PIN/Chiffrieren	j	n	n
PIN/Generieren	j	n	n
ICV/Zwischen-MAC ICV	n	n	n
Schlüsselteil/entfällt	j	n	n

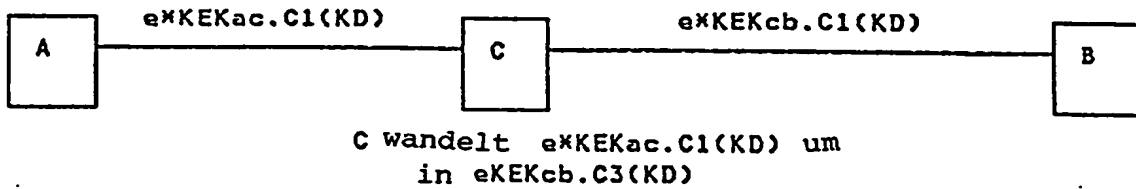
HINWEIS: Ein Schlüssel der Art/Unterart = Daten/Kompatibilität (d.h. ein Kompatibilitäts-Datenschlüssel) kann über einen CV-Kanal mit einem Steuervektor oder über einen CV=0-Kanal durch Entfernen des Steuervektors übertragen werden.

FIG. 71.

Modus (in Anweisung angegeben)	Verb.-Steuerung (in CV von KEK)	Kombination gültig?
CV	CV	ja
CV	CV=0	nein
CV	CV oder CV=0	ja
CV	entfällt	nein
CV=0	CV	nein
CV=0	CV=0	ja
CV=0	CV oder CV=0	ja
CV=0	entfällt	nein

47/58

**FIG. 72.**



**FIG. 73.**

Label1, C1L, C1R,  $e*KM.C1L(KEK1L)$ ,  $E*KM.C1R(KEK1R)$   
Label2, C2L, C2R,  $e*KM.C2L(KEK2L)$ ,  $E*KM.C2R(KEK2R)$   
Label3, C3L, C3R,  $e*KM.C3L(KEK3L)$ ,  $E*KM.C3R(KEK3R)$   
.  
Labeln, CnL, CnR,  $e*KM.CnL(KEKnL)$ ,  $E*KM.CnR(KEKnR)$

**FIG. 74.**

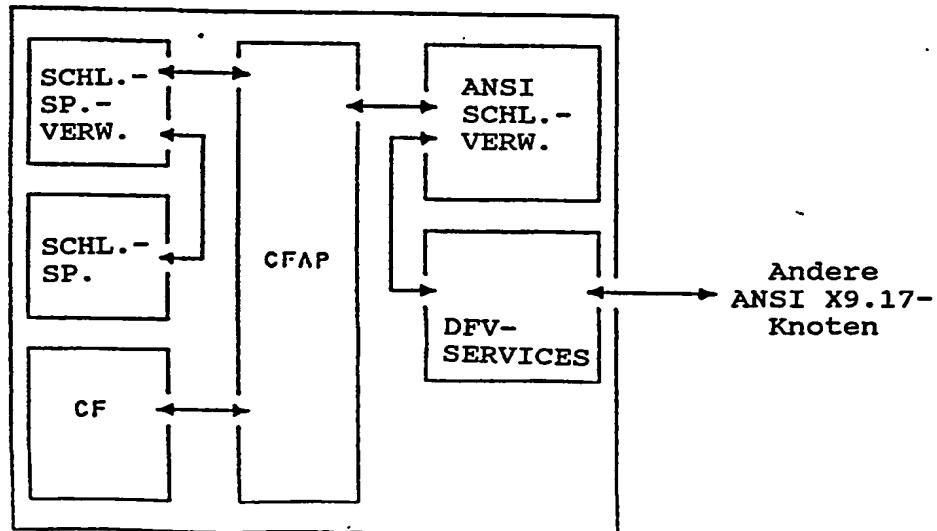


FIG. 75.

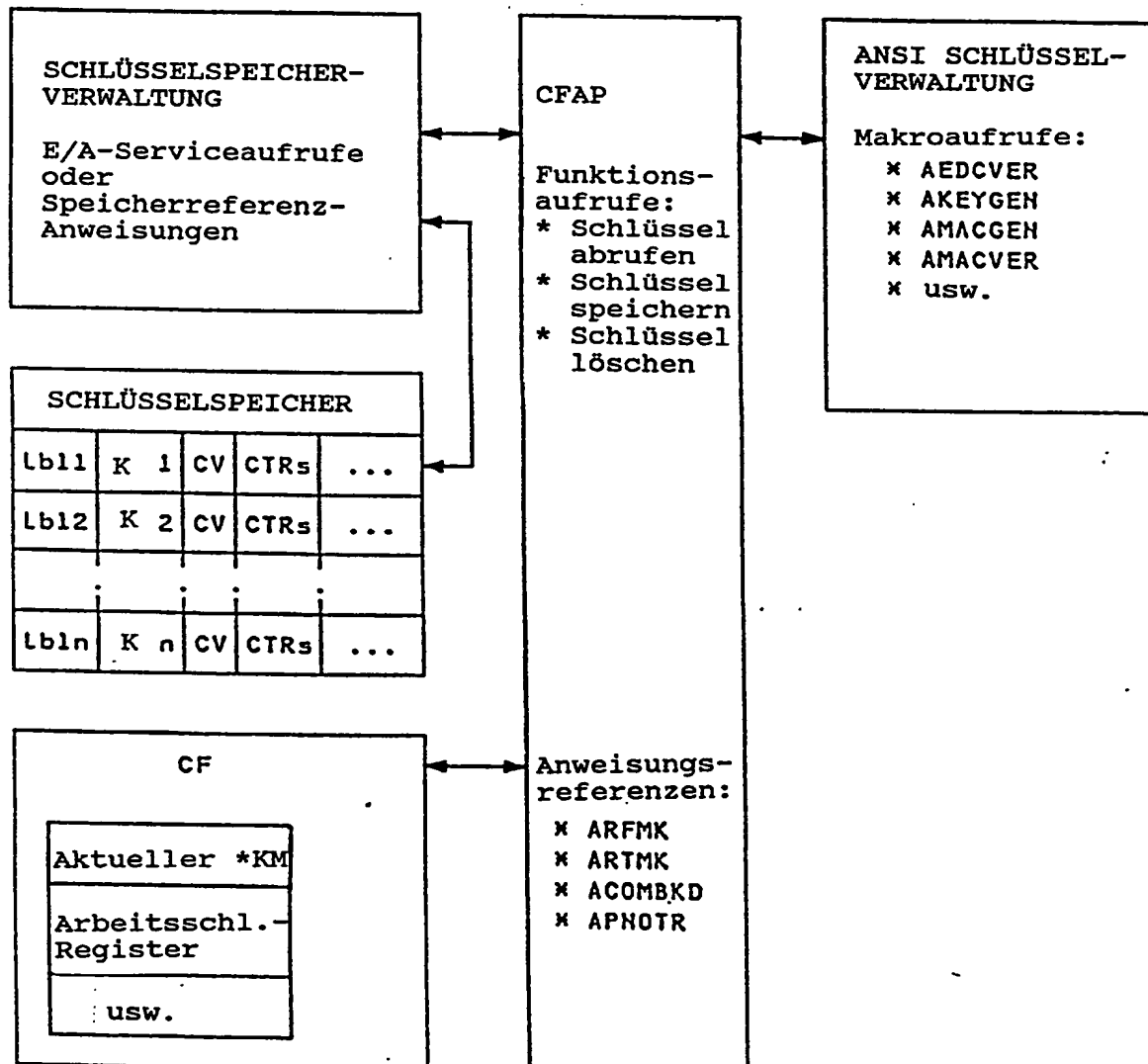


FIG. 77.

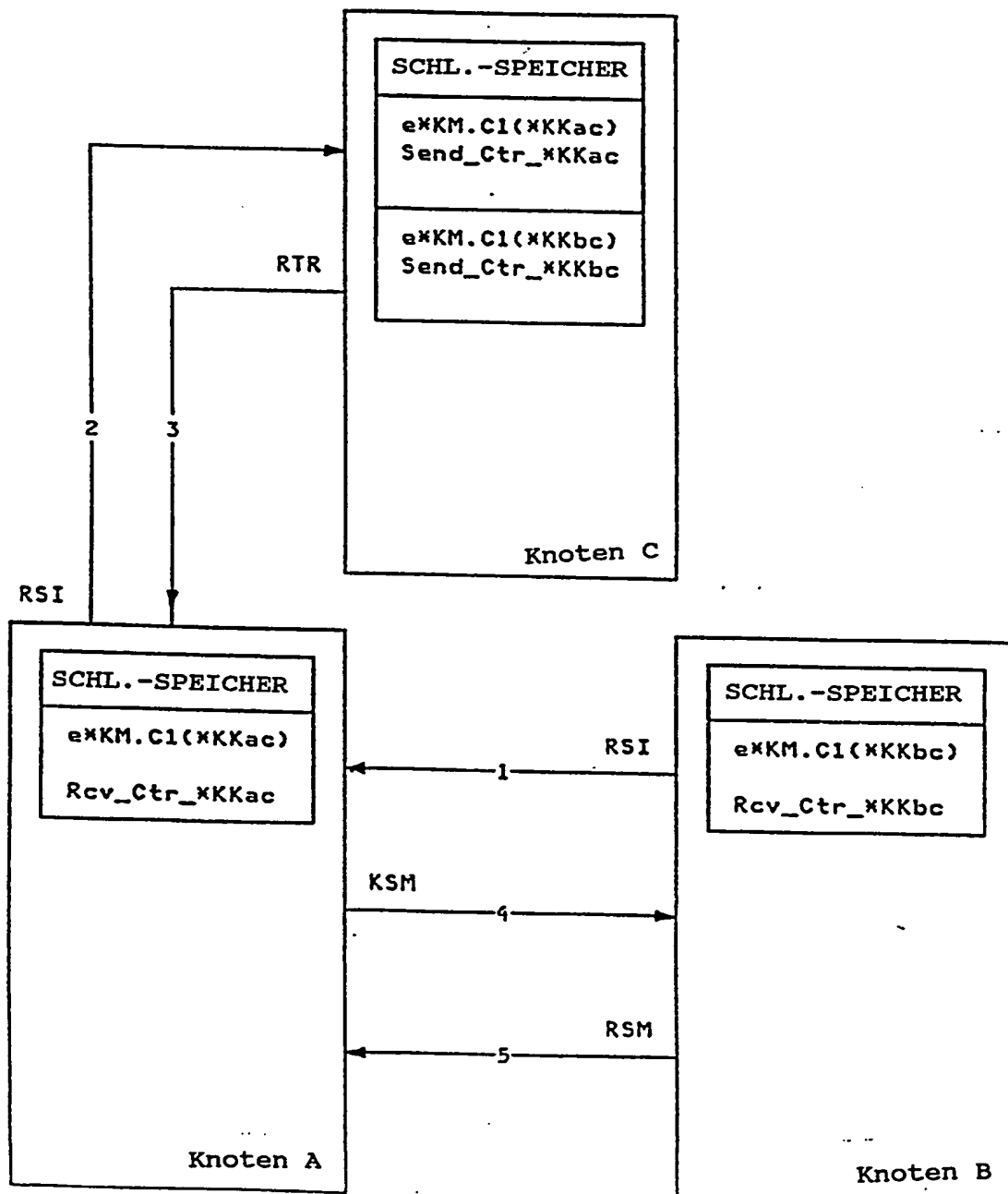
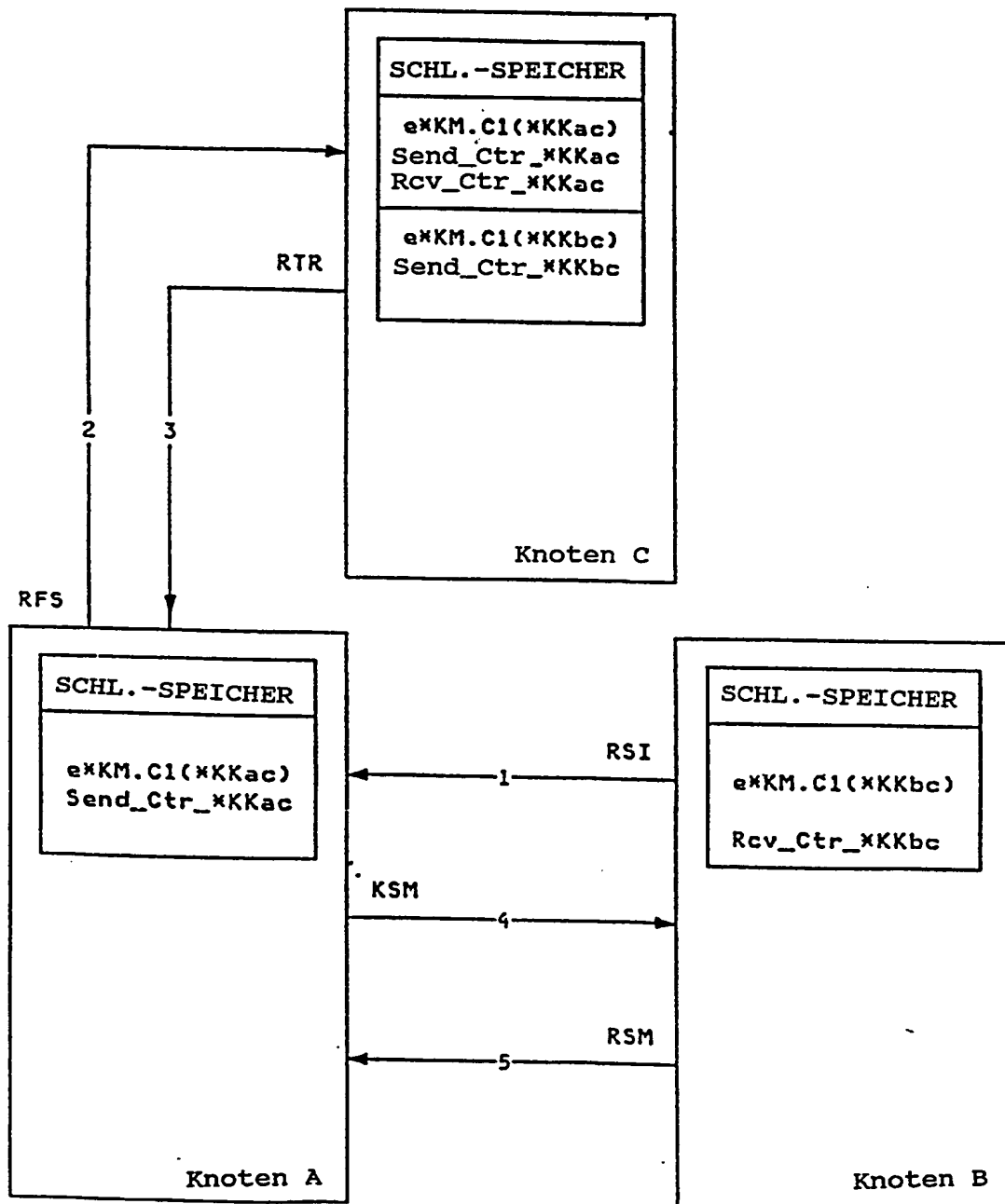


FIG. 78.



51/58

FIG. 79.

ENV	NOD	RCVS	FROM	UNDR <sup>1</sup>	GENS	STORES	UNDR <sup>2</sup>	SNDS	TO	UNDR <sup>3</sup>	SUP?
KTC	A				KK	KK  KK	*KMa	KK	KTC	*KKac	NEIN
					*KK	*KK	*KMa	*KK	KTC	*KKac	JA
	KTC	KK	A	*KKac		KK  KK (temp)	*KM <sub>c</sub>	KK	B via A	*KNcb	JA
		*KK	A	*KKac		*KK (temp)	*KM <sub>c</sub>	*KK	B via A	*KNcb	JA
	B	KK	KTC via A	*KNcb		KK  KK	*KM <sub>b</sub>				JA
		*KK	KTC via A	*KNcb		*KK	*KM <sub>b</sub>				JA
KDC	E N T F Ä L L T										
P-P	A				KK	KK  KK	*KMa	KK	B	KKab	NEIN
										*KKab	NEIN
										KNab	NEIN
										*KNab	NEIN
					*KK	*KK	*KMa	*KK	B	*KKab	JA
										*KNab	JA
	B			KKab							JA
				*KKab							JA
		KK	A	KNab		KK  KK	*KM <sub>b</sub>				JA
				*KNab							JA
				*KKab							JA
		*KK	A	*KNab		*KK	*KM <sub>b</sub>				JA



FIG. 76.

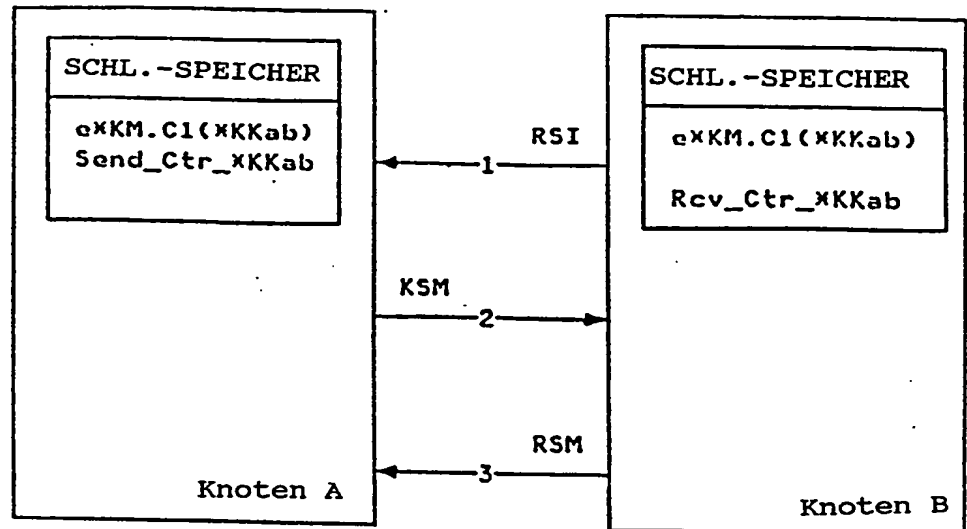


FIG. 80.

ENV	NOD	RCVS	FROM	UNDR <sup>1</sup>	GENS	STORES	UNDR <sup>2</sup>	SNDS	TO	UNDR <sup>3</sup>	SUP?
P-P	A				KD	KD	*KMa	KD	B	neuer KKab	NEIN
										neuer *KKab	JA
										KKab	JA
										*KKab	JA
										KHab	JA
										*KNab	JA
	B	KD	A	neuer KKab		KD	*KMb				JA
				neuer *KKab							JA
				KKab							JA
				*KKab							JA
				KHab							JA
				*KNab							JA

FIG. 81.

ENV	NOD	RCVS	FROM	UNDR <sup>1</sup>	GENS	STORES	UNDR <sup>2</sup>	SNDS	TO	UNDR <sup>3</sup>	SUP?
KDC	KDC				KD	KD (temp)	*KMc	KD	A und B via A	*KNca *KNcb	JA
	A	KD	KDC	*KNca		KD	*KMa				JA
	B	KD	KDC via A	*KNcb		KD	*KMb				JA
KTC	A				KDmac	KDmac (temp)	*KMa	KDmac	KTC	neuer KKab	NEIN
									neuer *KKab	JA	
					KD	KD	*KMa	KD	KTC	*KKac	JA
								B	neuer *KKab	JA	
	KTC	KDmac	A	neuer KKab		KDmac (temp)	*KMc				JA
				neuer *KKab							JA
	B	KD	A	*KKac		KD (temp)	*KMc	KD	B via A	*KNcb	JA
				neuer KKab		KD	*KMb				JA
				neuer *KKab							JA
				KTC via A			KD	*KMb			

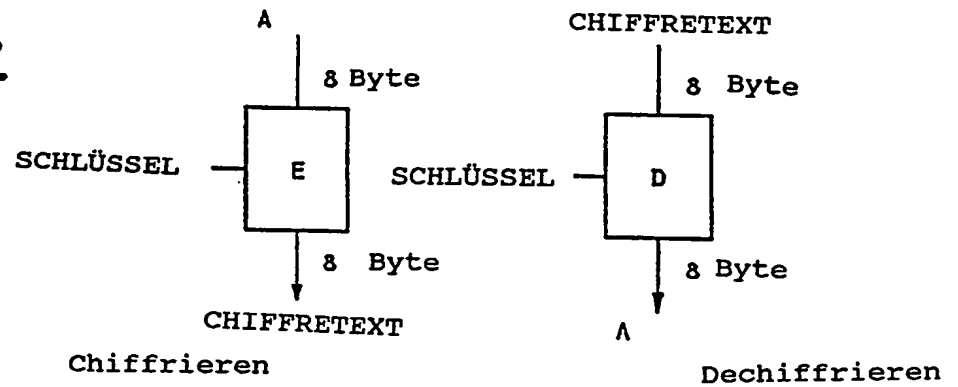
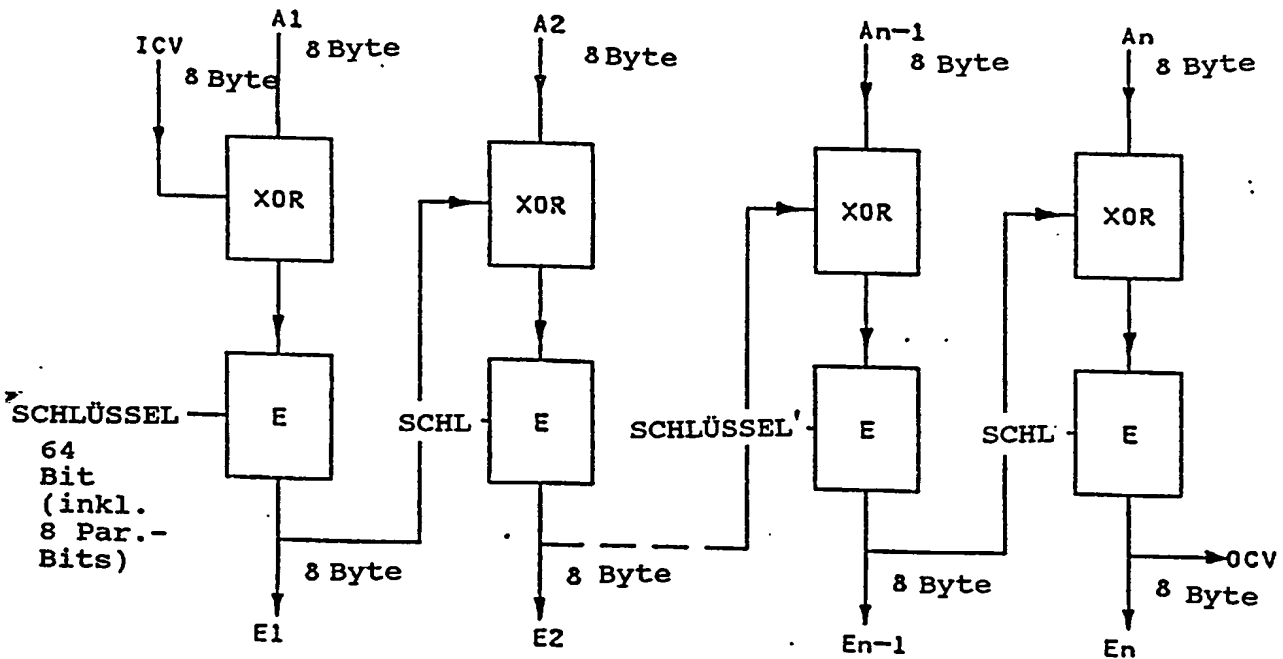
**FIG. 82.****FIG. 83.**

FIG. 84.

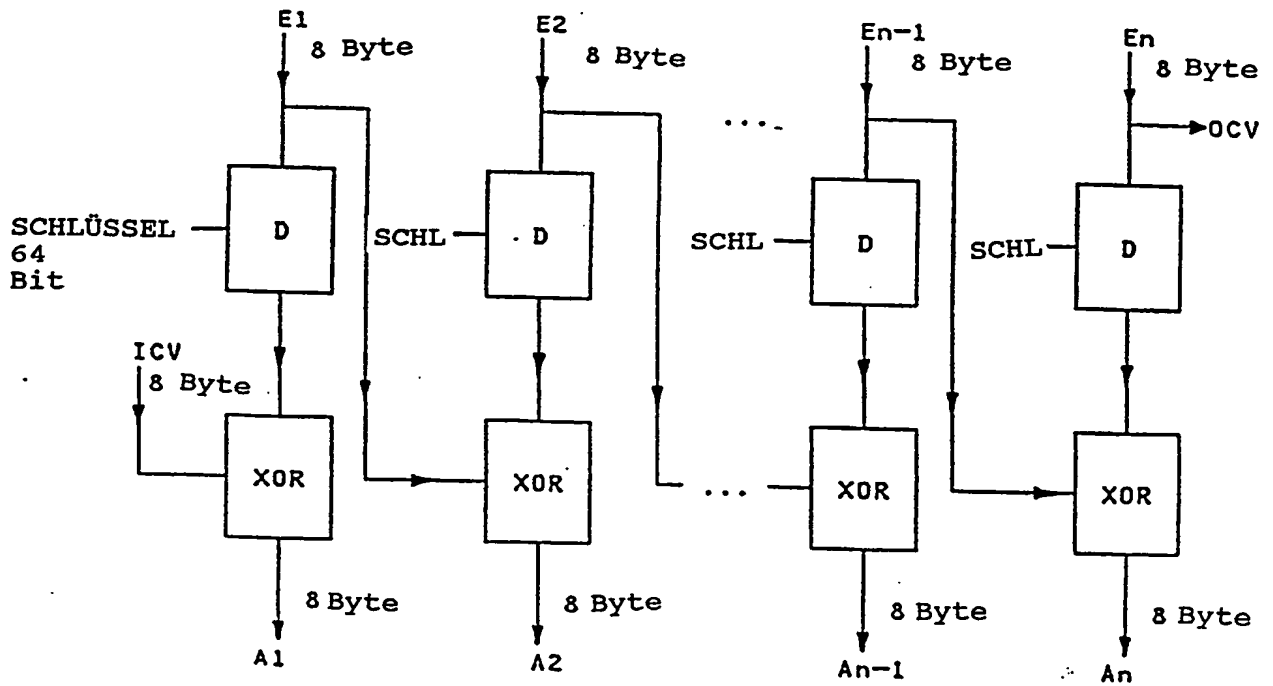
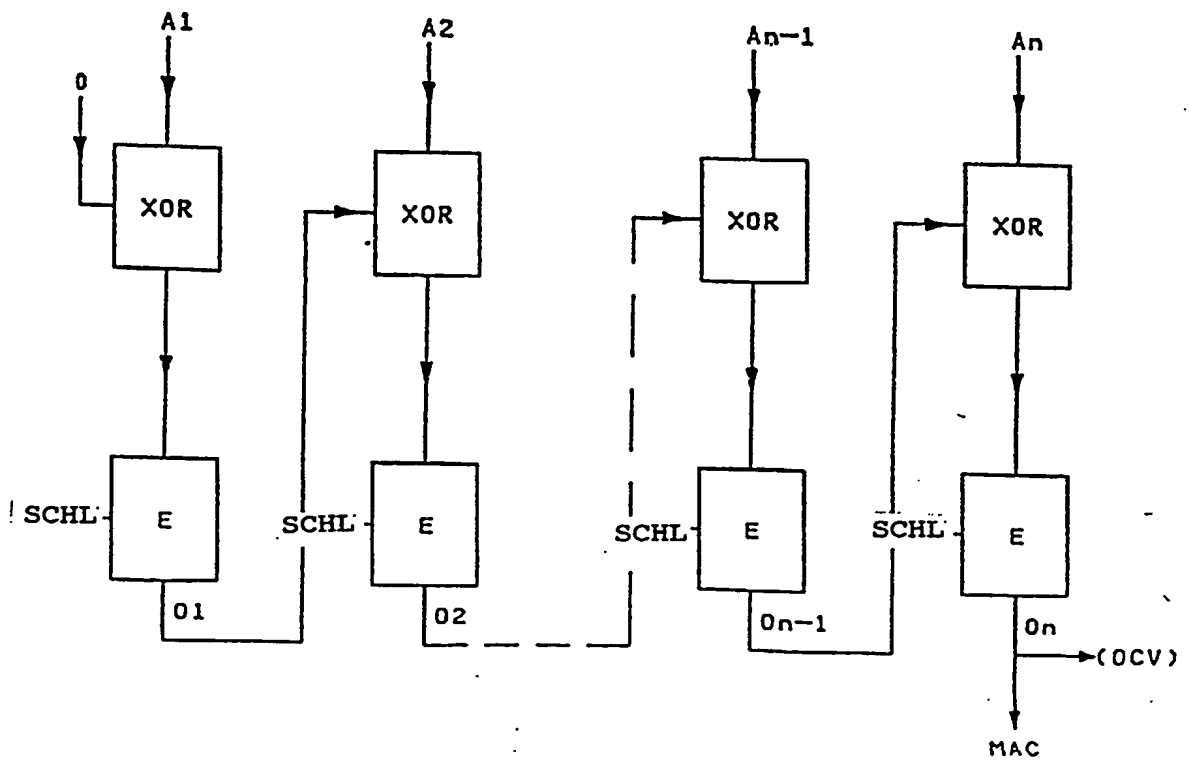


FIG. 85.



**FIG. 86.**

ANW.	GLEICHUNGEN
ENC	$KD, A \longrightarrow eKD(A)$ : EBC-Mod., 8 Byte D & Schl., codieren
DEC	$KD, eKD(A) \longrightarrow A$ : EBC-Mod., 8 Byte D & Schl., decod.
ENCI	$e*KM.C1(KD1), ICV, A, n, C1 \longrightarrow eKD1(ICV, A)$ chiff. lok.
DECI	$e*KM.C1(KD1), ICV, eKD1(ICV, A), n, C1 \longrightarrow A$ dechiff. lok.
GMAC	$e*KM.C1(KD1), <e*KM.C2(KD2)>, ICV <e*KM.C3(OCV)>, A, n,$ $ICV\text{-Art}, \text{Ausgabeart}, \text{mac-enc}, C1, <C2>, <C3>$ MAC(64 Bit) oder $e*KM.C3(OCV)$
VMAC	$e*KM.C1(KD1), <e*KM.C2(KD2)>, ICV <e*KM.C3(OCV)>, A, \text{MAC}, n,$ $ICV\text{-Art}, \text{Ausgabeart}, \text{mac-enc}, C1, <C2>, <C3>$ ja/nein oder $e*KM.C3(OCV)$
TCTXT	$e*KM.C1(KD1), ICV1, eKD1(ICV1, A), e*KM.C2(KD2), ICV2, n, C1, C2$ $\longrightarrow eKD2(ICV2, A)$

## FIG. 87.

ANW.	GLEICHUNGEN
GKS	$op-op : mode, C3, C4 \longrightarrow e*KM.C3(K), e*KM.C4(K)$ $op-ex : mode, C2L, C2R, C3, C4, e*KM.C2L(KKE2L),$ $e*KM.C2R(KKE2R) \longrightarrow e*KM.C3(K), e*KEK2.C4(K)$ $ex-ex : mode, C1L, C1R, C2L, C2R, C3, C4,$ $e*KM.C1L(KKE1L), e*KM.C1R(KKE1R),$ $e*KM.C2L(KKE2L), e*KM.C2R(KKE2R)$ $\longrightarrow e*KEK1.C3(K), e*KEK2.C4(K)$ $op-im : \text{wie bei } op-ex$ $im-ex : \text{wie bei } ex-ex$
RFMK	$dist-mode, e*KM.C1L(KEK1L), e*KM.C1R(KEK1R), e*KM.C2(K),$ $C1L, C1R, C2, C3 \longrightarrow e*KEK1.C3(K)$
RTMK	$dist-mode, e*KM.C1L(KEK1L), e*KM.C1R(KEK1R), e*KEK1.C3(K),$ $C1L, C1R, C2, C3 \longrightarrow e*KM.C2(K)$
KGEN	$Ausgabeart, C1 \longrightarrow K \text{ oder}$ $\longrightarrow e*KM.C1(K)$
EMK	$K, C1 \longrightarrow e*KM.C1(K)$
XLTKEY	$e*KM.C1L(KEK1L), e*KM.C1R(KEK1R),$ $e*KM.C2L(KEK2L), e*KM.C2R(KEK2R), e*KEK1.C3(K),$ $C1L, C1R, C2L, C2R, C3 \longrightarrow e*KEK2.C3(K)$
RTNMK	$mode, e*KMC.C1(K), C1 \longrightarrow e*KMN.C1(K)$
RTCMK	$mode, e*KMO.C1(K), C1 \longrightarrow e*KMC.C1(K)$
SMK	$() \longrightarrow NMK$
MDCOP	—

FIG. 88.

ANW.	GLEICHUNGEN
CLRCF	löscht alle Register in der CF
CLRKP	löscht das Schlüsselteile-Register
CLRNMK	löscht das Register für den neuen Hauptschlüssel
LCVA	$e*KM.C1(K), C1, C2 \longrightarrow e*KM.C2(K)$
LFMKP	$() \longrightarrow$ KP -Register wird in NMK-Register geladen
CMKP	$mode \longrightarrow$ NMK-Reg. = NMK-Reg. XOR Schlüsselteile-Reg.
LFKP	—
CKP	—
CVP	—
APHOTR	$mode, e*KM.C1L(KK1), e*KM.C1R(KK1), FMID, TOID, C1L, C1R, C2L, C2R$ $\longrightarrow e*KM.C2L(KKN1L), e*KM.C2R(KKN1R)$
ARFMK	$e*KM.C1L(KKL <KKN1L>), e*KM.C1R(KKR <KKN1R>),$ $e*KM.C2(SCHL), Zähler, C1L, C1R, C2 \longrightarrow e*KK()$
ARTMK	$e*KM.C1L(KKL <KKN1L>), e*KM.C1R(KKR <KKN1R>),$ $e*KK(SCHL), Zähler, Schlüsselart, C1L, C1R, C2, <C3>$ $\longrightarrow e*KM.C2(SCHL), e*KM.C3(SCHL)$ (für KD) $\longrightarrow e*KM.C2(SCHL),$ (für KK)
AXLTKEY	$e*KM.C1L(KK1L <KKN11L>), e*KM.C1R(KK1R <KKN11R>),$ $e*KM.C2L(KK2L <KKN12L>), e*KM.C2R(KK2R <KKN12R>),$ $e*KK1(SCHL), Zähler1, Zähler2, Schl.-Art, C1L, C1R, C2L, C2R, C3$ $\longrightarrow e*KK2(SCHL), e*KM.C3(SCHL)$ (für KD) $\longrightarrow e*KK2(SCHL), e*KM.C3(KDmac)$ (für KK)
ACOMBKD	$e*KM.C1(KD1), e*KM.C2(KD2), C1, C2, C3 \longrightarrow e*KM.C3(KD)$




[Subscribe](#) (Full Service) [Register](#) (Limited Service, Free) [Login](#)

Search: ☐ The ACM Digital Library ☒ The Guide



## THE GUIDE TO COMPUTING LITERATURE

 [Feedback](#) [Report a problem](#) [Satisfaction survey](#)

### Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Faults

**Source** [Lecture Notes In Computer Science; Vol. 1361](#) [archive](#)  
[Proceedings of the 5th International Workshop on Security Protocols](#) [table of contents](#)  
Pages: 115 - 124  
Year of Publication: 1997  
ISBN:3-540-64040-1

**Authors** [Feng Bao](#)  
[Robert H. Deng](#)  
[Yongfei Han](#)  
[Albert B. Jeng](#)  
[A. Desai Narasimhalu](#)  
[Teow-Hin Ngair](#)

**Publisher** Springer-Verlag London, UK

**Additional Information:** [citations](#) [collaborative colleagues](#)

**Tools and Actions:** [Find similar Articles](#) [Review this Article](#)  
[Save this Article to a Binder](#) Display Formats: [BibTex](#) [EndNote](#) [ACM Ref](#)

#### ↑ CITINGS 5

[Feng Bao, Multimedia content protection by cryptography and watermarking in tamper-resistant hardware, Proceedings of the 2000 ACM workshops on Multimedia, p.139-142, October 30-November 03, 2000, Los Angeles, California, United States](#)

[Johannes Blömer, Martin Otto, Jean-Pierre Seifert, A new CRT-RSA algorithm secure against bellcore attacks, Proceedings of the 10th ACM conference on Computer and communications security, October 27-30, 2003, Washington D.C., USA](#)

[Mathieu Ciet, Marc Joye, Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults, Designs, Codes and Cryptography, v.36 n.1, p.33-43, July 2005](#)

[Sung-Ming Yen, Marc Joye, Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis, IEEE Transactions on Computers, v.49 n.9, p.967-970, September 2000](#)

[Sung-Ming Yen, Seungjoo Kim, Seongan Lim, Sang-Jae Moon, RSA Speedup with Chinese Remainder Theorem Immune against Hardware Fault Cryptanalysis, IEEE Transactions on Computers, v.52 n.4, p.461-472, April 2003](#)

#### ↑ Collaborative Colleagues:

Feng Bao:	Sabine R. Öhring	Yongfei Han	Yi Mu	Zhiguo Wan
	Tatsuya Akutsu	Jian Hu	A. Desai	Guilin Wang
	Robert Deng	Y. Igarashi	Narasimhalu	Hongjun Wu
	Robert Deng	Yoshihide	Teow-Hin Ngair	Yongdong
	Robert H. Deng	Igarashi	Khanh Quoc	Wu
	Gui-Liang Feng	Y. Igarashi	Nguyen	Yanjiang
	Peirong Feng	Yukihiro Iwasaki	Koji Obokata	Yang
	Y. Funyu	Albert B. Jeng	HweeHwa Pang	Ding-Feng



# Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults

F. Bao, R. H. Deng, Y. Han, A. Jeng, A. D. Narasimhalu, T. Ngair  
Institute of Systems Science  
National University of Singapore  
{baofeng, deng, yfhan, jeng, desai, teowhin}@iss.nus.sg

**Abstract.** In this paper we present a method of attacking public-key cryptosystems (PKCs) on tamper resistant devices. The attack makes use of transient faults and seems applicable to many types of PKCs. In particular, we show how to attack the RSA, the ElGamal signature scheme, the Schnorr signature scheme, and the DSA. We also present some possible methods to counter the attack.

## 1 Introduction

In September 1996, Boneh, DeMillo and Lipton from Bellcore announced a new type of cryptanalytic attack against RSA-like public key cryptosystems on tamper resistant devices such as smart card [4]. However, technical details of the Bellcore attack were withheld in that announcement and was released only at the end of October 1996. On 18th October 1996, Biham and Shamir published their attack, called Differential Fault Analysis (DFA), to secret key cryptosystems [5], such as DES. Some concrete ideas on how their attack works were revealed in their announcement.

Our work here was motivated first by the Bellcore announcement and then by the DFA announcement. Our first report on attacking RSA and some countermeasures were posted in the Internet on the 23rd and 24th October 1996 [2]. Right after that, A. K. Lenstra sent us his memo [9] on attacking RSA in Chinese remainder in a private communication. Subsequently, we released a more complete research note on attacking RSA and the ElGamal signature scheme on the 29th October 1996 [3]. Recently, Joye and Quisquater extended the Chinese remaindering attack to LUC and Demytko cryptosystems [8].

In this paper, we continue our earlier effort of attacking public-key cryptosystems (PKCs) on tamper resistant devices. Our attacking model makes use of the transient faults and seems applicable to many types of PKCs, such as RSA-like schemes and discrete logarithm based schemes. As in the Bellcore and DFA announcements, we assume that by exposing a sealed tamper resistant device such as a smart card to certain physical effects (e.g., ionizing or microwave radiation), one can induce with reasonable probability faults at random bit locations in a tamper resistant device at some random intermediate stage in the cryptographic computation. The faults in the random bit locations do not influence the code

itself, i.e., the program itself does not crash, and only some of the values it operates upon are affected. It is further assumed that the attacker is in physical possession of the tamper resistant device and that he can repeat the experiment with the same private key by applying external physical effects to obtain faulty outputs.

The organization of the paper is as follows. In Section 2, we first report our attacks to RSA, and then present Lenstra's attack to RSA implemented based on the Chinese Remainder Algorithm (CRA). In Section 3, we show how to break discrete logarithm based schemes such as the ElGamal signature scheme [7], the Schnorr signature scheme [11], and the Digital Signature Algorithm (DSA). At the end of each section, we also give some possible methods to counter the attack.

When this manuscript was near its completion, Dan Boneh kindly sent us their paper [6] in a private communication. Throughout this paper, we will make remarks about the relation between their paper and our work wherever it is appropriate.

## 2 Attacking the RSA Scheme

Let  $n = pq$  be the product of two primes  $p$  and  $q$  in RSA,  $e$  be the public exponent which is publicly known and  $d$  be the private exponent which is stored inside the tamper resistant device. Our attacks to RSA will be described in terms of ciphertext decryption although they can also be described in terms of signature generation.

Let  $m$  be a plaintext, then the corresponding ciphertext is

$$c \equiv m^e \pmod{n}$$

Denote the binary representation of the private exponent as  $d_{t-1}|d_{t-2}|\dots|d_i|\dots|d_1|d_0$ , where  $d_i$ , taking value 1 or 0, is the  $i$ th bit,  $t$  is the number of bits in  $d$ , and  $x|y$  denotes concatenation of  $x$  and  $y$ . Further, we denote

$$c_i \equiv c^{2^i} \pmod{n}, \text{ for } i = 0, 1, 2, \dots, t-1$$

Given  $c$  and  $d$ , the corresponding plaintext  $m$  can be expressed as

$$m \equiv c^d \pmod{n} \equiv c_{t-1}^{d_{t-1}} \dots c_i^{d_i} \dots c_1^{d_1} c_0^{d_0} \pmod{n}$$

### 2.1 Attack I

For the sake of simplicity, here we assume that in decrypting a ciphertext a single bit error is induced in  $c_i$ , for a random  $i \in \{0, 1, 2, \dots, t-1\}$ . Denote the corrupted value as  $c'_i$ . Then the output from the tamper resistant device is

$$m' \equiv c_{t-1}^{d_{t-1}} \dots c_i'^{d_i} \dots c_1^{d_1} c_0^{d_0} \pmod{n}$$

The attacker now has both  $m$  and  $m'$  so that he is able to compute

$$\frac{m'}{m} \equiv \frac{c_i'^{d_i}}{c_i^{d_i}} \mod n$$

which equals  $\frac{c_i'}{c_i} \mod n$  if  $d_i = 1$  or equals 1 if  $d_i = 0$ . (From now on we assume that every number we meet is relatively prime with respect to  $n$ , hence we can compute its inverse.) The attacker can easily compute all the possible  $\frac{c_i'}{c_i} \mod n$  values in advance (there are a total of  $t^2$  such values since  $c_i'$  has  $t$  possible values). Now the attacker compares all these values with  $\frac{m'}{m} \mod n$ . Once a match is found, he knows  $i$  and then knows that  $d_i$  is 1.

This simple example is just meant to illustrate the basic ideas of our attack. It showed that one bit fault at certain location and time can cause fatal leakage of the private key.

The example above assumes that only one  $c_i$  contains a single bit error and that there is no error propagation from  $c_i$  to  $c_j$ ,  $j > i$ . The effects of such error propagation were considered in [6] and [9]. As a result, the error models in [6] and [9] are more complicated and probably more realistic than ours. From practical viewpoint, our model can be explained as the model for “read” error. That is,  $c_i$  is mistaken as  $c_i'$  when it is multiplied to the value for computing  $c^d$  but remains correct when it is squared to obtain  $c_{i+1}$ .

Another issue is that we can actually consider multi-bit faults instead of one bit fault only. In this case, we need to compare  $m'/m \mod n$  with many more possible values. For the case of two-bit faults,  $m'/m \mod n$  should be matched with all the values  $c_{i_1}' c_{i_2}' / c_{i_1} c_{i_2} \mod n$  ( $i_1, i_2 \in \{0, 1, 2, \dots, t-1\}$ ) and  $c_i'' / c_i \mod n$ , where  $c_i''$  denotes the value of two bit errors in  $c_i$ . In this case,  $O(t^4)$  possible values should be generated in advance and matched (as well as those  $c_i' / c_i \mod n$ ) with the value  $m'/m \mod n$ . In general, about  $t^{2j}$  values need to be generated in the situation where  $j$ -bit faults may take place.

## 2.2 Attack II

Suppose that one bit in the binary representation of  $d$  is flipped and that the faulty bit position is randomly located. An attacker arbitrarily chooses a plaintext  $m$  and computes the ciphertext  $c$ . He then asks the tamper resistant device to decrypt  $c$  and induces a random bit error in  $d$  by applying external physical effects to the device. Assuming that  $d_i$  is changed to its complement  $d_i'$ , then the output of the device will be

$$m' \equiv c_{t-1}^{d_{t-1}} \cdots c_i^{d_i'} \cdots c_1^{d_1} c_0^{d_0} \mod n$$

Since the attacker now possesses both  $m$  and  $m'$ , he can compute

$$\frac{m'}{m} \equiv \frac{c_i^{d_i'}}{c_i^{d_i}} \mod n.$$

Obviously, if  $m'/m \equiv 1/c_i \pmod n$ , then  $d_i = 1$ , and if  $m'/m \equiv c_i \pmod n$ , then  $d_i = 0$ . Therefore, the attacker can compare  $m'/m \pmod n$  to  $c_i \pmod n$  and  $c_i^{-1} \pmod n$ , for  $i = 0, 1, \dots, t-1$ , in order to determine one bit of  $d$ . He repeats the above process using either the same plaintext/ciphertext pair or using different plaintext/ciphertext pairs until enough information in  $d$  is obtained.

Suppose one bit error takes place randomly in  $d$  in each fault test. Then by basic probabilistic counting, we have the following: If we take  $t \log t$  fault tests, with a probability larger than half, every bit of  $d$  is disclosed.

It should be noted again that this attack applies to the case of multiple bit errors. Assuming two bit faults. The attacker needs to compare  $m'/m \pmod n$  with  $c_i c_j \pmod n$ ,  $c_i/c_j \pmod n$ , and  $1/(c_i c_j) \pmod n$ , for all  $i, j \in \{0, 1, 2, \dots, t-1\}$ . In this case, matching  $m'/m \pmod n$  with all these values has a complexity of  $O(t^2)$  instead of  $O(t)$  as in the single error case; while with large possibility one obtain two bits,  $d_i$  and  $d_j$ , once a successful match is obtained.

### 2.3 Lenstra's Attack on RSA with Chinese Remainder Algorithm

Boneh, DeMillo and Lipton [6] gave an attack on RSA implemented with the CRA. Their attack requires two signatures of a given message: one correct signature and one faulty signature. Lenstra independently worked out a similar attack against RSA with CRA which requires only one faulty signature of a known message [9]. In the following, we briefly outline Lenstra's attack.

The signature  $s$  of a message  $m$  equals  $m^d \pmod n$  and thus  $s^e \pmod n$  is again equal to  $m \pmod n$ . It is well known that  $s$  can be computed by computing

$$u \equiv m^d \pmod p \text{ and } v \equiv m^d \pmod q,$$

and by combining  $u$  and  $v$  using the CRA. If a fault occurs in the course of the computation of the signature, the resulting value, denoted as  $s'$ , will most likely not satisfy  $m \equiv s'^e \pmod n$ . If, however, the fault occurred only during the computation of say,  $u$ , and if  $v$  and the CRA were carried out correctly, then the resulting faulty signature  $s'$  satisfies  $s'^e \equiv m \pmod q$ , but the same congruence  $\pmod p$  does not hold. Therefore,  $q$  divides  $s'^e - m$  but  $p$  does not divide  $s'^e - m$ , so that a factor of  $n$  may be discovered by the recipient of the faulty signature  $s'$  by computing the greatest common divisor of  $n$  and  $s'^e - m$ . This attack is very powerful since it requires only one faulty signature and it works under a general fault model.

### 2.4 Some Possible Countermeasures

There may be a variety of attacks to PKCs by inducing faults. The means of breaking a PKC can be devised to be dependent on the specific PKC algorithm as well as on its implementation. Generally speaking, countermeasures to such attacks are relatively insensitive to both the implementation of a PKC and the attacking scenarios. Here we envisage two general approaches to counter such attacks, one is based on the principle of "check and balance" and the other

based on the principle of “information hiding”. The former can be done by checking/verifying the result before sending it to the outside world and the latter can be achieved by introducing some randomness in the intermediate stages of the cryptographic computation.

- a) The attacks may be avoided by calculating the output 2 times and matching the two results. However, this approach doubles the computational time. As pointed out in [4], this double computation method also avoids their attack. The weakness of this counter measure is that it slows down the computation by a factor of 2, which is “not accepted for some applications” [4].
- b) In many cases, the encryption key  $e$  is usually small. So we can verify the result by checking  $m'^e = c \bmod n$ ? It is much more efficient than the double computation approach if  $e$  is small. This approach was also pointed out independently by Lenstra.
- c) In some protocols for digital signature, a random string is chosen by the smart card and concatenated to a message  $m$  which is to be signed by the smart card. For example,  $m$  is a 412 binary string given to the smart card. The smart card randomly chooses a 100 bit number  $r$  and the output is  $(m|r)^d \bmod n$ . Since  $r$  is different each time, the attack does not work in such case.
- d) In the case where  $e$  is large and where the tamper resistant device is required to compute  $c^d \bmod n$ , the following efficient method may be used to counter the attack. The tamper resistant device generates a random number  $r$  and computes  $r^d \bmod n$ . This can be done in advance, i.e., before  $c$  is input and when the device is idle. To compute  $c^d \bmod n$ , the device first computes  $rc \bmod n$ , then  $(rc)^d \bmod n$ , and finally  $\frac{(rc)^d}{r^d} \bmod n$ . If no fault takes place, the output is obviously correct. If any fault takes place, the output is masked by  $r$ . Since  $r$  is unknown to the attacker and different for every decryption, our attack does not work. For the example, in the case of Attack II, if  $d_i$  is 0 and  $d'_i$  is 1, then  $m'/m \equiv r^{2^i} c_i \bmod n$ . Since  $r$  is unknown to the attacker, the ratio is useless to him.

It should be pointed out that a) - d) work against our attacks while only a) - c) work against Lenstra’s attack.

### 3 Attacking Discrete Logarithm Based Schemes

The general concept of attacking the RSA scheme can be applied to attack against discrete logarithm based public key cryptosystems. In the following, we show our attacks to the ElGamal signature scheme, the Schnorr signature scheme, and the DSA. Throughout this section, we will denote a signer’s private key as  $x$  and its binary representation as  $x_{t-1}|x_{t-2}|\cdots|x_i|\cdots|x_1|x_0$ , where  $t$  is the number of bits in  $x$  and  $x_i$  is the  $i$ th bit of  $x$ . The private key is kept inside a tamper resistant device and the corresponding public key can be made available to everyone.

The general steps followed by an attacker are as follows: 1) the attacker applies external physical effects to induce some bit errors at random locations in  $x$  and then obtains a faulty signature, 2) he performs some computations on the faulty signature to uncover part of the private key  $x$ . The attacker repeats steps 1) and 2) until he uncovers the binary representation of  $x$  or a sufficiently large number of bits that allow him to discover the rest of  $x$  by brute force. To keep the paper compact, we will only show steps 1) and 2) in the following, without explicitly showing the loops of the attack.

To simplify the description, we first show the attacks for the case of single bit error. We then briefly discuss the case of multiple bit errors.

### 3.1 Attacking the ElGamal Signature Scheme

In the ElGamal signature scheme [7], to generate a private and public key pair, we first choose a prime  $p$ , and two random numbers,  $g$  and  $x$ , such that both  $g$  and  $x$  are less than  $p$ . The private key is  $x$  and the public key is  $(y \equiv g^x \bmod p, g, p)$ .

To generate a signature on a message  $m$ , the signer first picks a random  $k$  such that  $k$  is relatively prime to  $p - 1$ . She then computes

$$w \equiv g^k \bmod p \text{ and } s \equiv (m - xw)/k \bmod (p - 1)$$

The signature is the pair  $w$  and  $s$ . To verify the signature, the verifier confirms that

$$y^w w^s \equiv g^m \bmod p.$$

Assume that  $x_i$  in  $x$  is changed to its complement  $x'_i$  during the process of signing of a message  $m$ . We denote the corrupted  $x$  as  $x'$  due to the flip of  $x_i$ . Then the outputs of the device will be

$$w \equiv g^k \bmod p \text{ and } s' \equiv (m - x'w)/k \bmod (p - 1)$$

Using  $w, s', m$ , and the signer's public key  $(y, p, g)$ , the attacker computes

$$T \equiv y^w w^{s'} \bmod p \equiv g^m g^{w(x-x')} \bmod p.$$

Let  $R_i \equiv g^{w2^i} \bmod p$  for  $i = 0, 1, 2, \dots, t - 1$ . Then, we have

$$TR_i \equiv g^m \bmod p, \text{ if } x_i = 0$$

(since for  $x_i = 0$  we have  $x - x' = -2^i$ ) and

$$\frac{T}{R_i} \equiv g^m \bmod p, \text{ if } x_i = 1$$

(since for  $x_i = 1$  we have  $x - x' = 2^i$ ). The attacker computes  $TR_i$  and  $T/R_i$  and tests to see if either  $TR_i$  or  $T/R_i$  equals  $g^m \bmod p$ , for  $i = 0, 1, \dots, t - 1$ . If a match is found, then one bit of  $x$  is found.

### 3.2 Attacking the Schnorr Signature Scheme

In the Schnorr signature scheme [11], to generate a private and public key pair, we first choose two primes,  $p$  and  $q$ , such that  $p = zq + 1$  for a reasonably large  $q$ . We then select a number  $g$  not equal to 1, such that  $g^q \equiv 1 \pmod{p}$ . The signer's private key is a random  $x$  less than  $q$ , and the public key is  $(y \equiv g^{-x} \pmod{p}, g, p, q)$ .

To generate a signature on a message  $m$ , the signer first picks a random  $k$  that is less than  $q$ . She then computes

$$w \equiv g^k \pmod{p}, e = h(m|w) \text{ and } s \equiv ex + k \pmod{q},$$

where  $h$  is a secure one-way hash function that outputs a number less than  $q$ . The signature is the pair  $e$  and  $s$ . Because

$$(g^s y^e \pmod{p}) = w,$$

to verify the signature, the verifier confirms that

$$h(m|(g^s y^e \pmod{p})) = e$$

During the computation of  $s$ , assuming that  $x_i$  in  $x$  is flipped to  $x'_i$  and denote the corrupted  $x$  as  $x'$ . Then the outputs of the device will be

$$e \equiv h(m|w) \pmod{p} \text{ and } s' \equiv ex' + k \pmod{q}$$

Using  $e, s', m$ , and the signer's public key  $(y, p, g, q)$ , the attacker computes

$$T \equiv g^{s'} y^e \equiv w g^{e(x' - x)} \pmod{p}$$

Let  $R_i \equiv g^{e2^i} \pmod{p}$  for  $i = 0, 1, 2, \dots, t-1$ . It is easy to see that  $TR_i \equiv w g^{e(x' - x + 2^i)} \pmod{p}$  and  $T/R_i \equiv w g^{e(x' - x - 2^i)} \pmod{p}$ . Then we have

$$h(m|(TR_i \pmod{p})) = e, \text{ if } x_1 = 1$$

(since for  $x_1 = 1$ , we have  $x' - x = -2^1$  and then  $TR_i \equiv w \pmod{p}$ ), and

$$h(m|(T/R_i \pmod{p})) = e, \text{ if } x_i = 0$$

(since for  $x_i = 0$ , we have  $x' - x = 2^i$  and then  $T/R_i \equiv w \pmod{p}$ ). Therefore, by iterating through different  $i$  and matching  $e$  with  $h(m|(TR_i \pmod{p}))$  and  $h(m|(T/R_i \pmod{p}))$ , the attacker can discover the  $i$ th bit,  $x_i$ , of the private key  $x$ .

In [6], Boneh, DeMillo and Lipton gave an attack against the Schnorr identification scheme [11]. In their attack, it was required that 1) the verifier uses the same challenge  $e$  (which plays the same role as the  $e$  in the Schnorr signature scheme) in all invocations of the identification protocol without being detected by the prover's tamper resistant device, and 2) a bit error is introduced in the random number  $k$  (which plays the same role as the  $k$  in the Schnorr signature scheme). Because of these two requirements, their attack can not be applied to break the Schnorr signature scheme. On the other hand, our attack to the Schnorr signature scheme can be applied to break the Schnorr identification scheme with little modification.

### 3.3 Attacking the DSA

In the DSA, to generate a private and public key pair, we first choose a prime  $p$  such that  $p = zq + 1$  for a reasonably large prime  $q$ . We then compute  $g \equiv b^{(p-1)/q} \pmod{p}$ , where  $b$  is any number less than  $p-1$  such that  $(b^{(p-1)/q} \pmod{p})$  is greater than 1. The signer's private key is  $x$ , a random number less than  $q$ , and the public key is  $(y \equiv g^x \pmod{p}, g, p, q)$ .

To sign a message  $m$ , the signer first picks a random  $k$  that it is less than  $q$ . She then computes

$$w \equiv g^k \pmod{p \pmod{q}} \text{ and } s \equiv (e + wx)/k \pmod{q},$$

where  $e = h(m)$  with  $h$  being a secure one-way hash function that outputs a number less than  $q$ . The signature is the pair  $w$  and  $s$ . To verify the signature, the verifier confirms that

$$w \equiv g^{(ue \pmod{q})} y^{(uw \pmod{q})} \pmod{p \pmod{q}},$$

where  $u \equiv 1/s \pmod{q}$ .

The attacker applies external physical effects to the tamper resistant device and at the same time asks the device to sign a message  $m$ . During the process of calculating  $s$ , we assume that the  $i$ th bit of  $x$  is changed from  $x_i$  to its complement  $x'_i$ . Let  $x'$  denote the corrupted  $x$  due to the flip of  $x_i$ . Then the outputs of the device will be

$$w \equiv g^k \pmod{p \pmod{q}} \text{ and } s' \equiv (e + wx')/k \pmod{q}$$

Using  $w$ ,  $u' \equiv 1/s' \pmod{q}$ ,  $m$ , and the signer's public key  $(y, p, g, q)$ , the attacker can compute  $e = h(m)$  and

$$T \equiv g^{(u'e \pmod{q})} y^{(u'w \pmod{q})} \equiv g^{(u'(e+wx) \pmod{q})} \pmod{p \pmod{q}}.$$

Let  $R_i \equiv g^{(u'w2^i \pmod{q})} \pmod{p \pmod{q}}$  for  $i = 0, 1, 2, \dots, t-1$ . Then we have

$$TR_i \equiv g^{(u'(e+w(x+2^i)) \pmod{q})} \pmod{p \pmod{q}}$$

$$T/R_i \equiv g^{(u'(e+w(x-2^i)) \pmod{q})} \pmod{p \pmod{q}}.$$

It is easy to show that

$$TR_i \equiv w \pmod{p \pmod{q}}, \text{ if } x_i = 0 \quad T/R_i \equiv w \pmod{p \pmod{q}}, \text{ if } x_i = 1$$

So by iterating through different  $i$  and matching  $w$  with  $TR_i \pmod{p \pmod{q}}$  and  $T/R_i \pmod{p \pmod{q}}$ , the attacker can discover the value of  $x_i$ .

### 3.4 Multiple Bit Errors

Extension of the above methods in attacking discrete log based digital signature schemes to the cases of multiple faults in  $x$  is straightforward. In the case of single fault, the  $R$ s, denoted as  $R_i$  in the above single bit error case, each has a single argument  $i$ . Their computation and subsequent comparison is of complexity  $O(2t)$ . In the case of  $j > 1$  faults in  $x$ , the  $R$ s, which we will denote as  $R_{i_1, i_2, \dots, i_j}$ , will each have  $j$  arguments and their computations and subsequent comparisons are of complexity  $O((2t)^j)$ .



### 3.5 Countermeasures

The countermeasure achieved through double computation as mentioned in section 2.4 also applies to the attacks presented in this section.

Another countermeasure to the attack against discrete log based signature schemes is that the tamper resistant device stores both  $x$  and  $1/x$ , where  $x$  is used in the computation of  $s$  and  $1/x$  is used to check the correctness of  $s$ . As an example let's consider the Schnorr signature scheme. Right after computing  $s' \equiv ex' + k \bmod q$ , the device verifies the value of  $s'$  by comparing  $e$  with  $(s' - k)(1/x) \bmod q$ . If these two values are the same, the result is considered correct; otherwise, the device is reset.

In general, to prevent corrupted variables from being used in a calculation and subsequently causing breaking of a cryptosystem, we suggest that the variable and its inverse be stored somewhere before the calculation takes place. The variable is used for the calculation and its inverse can be used to verify the result of the calculation.

To illustrate the above concept, let's again consider the Schnorr signature scheme. Suppose we want to make sure that the correct values of both  $x$  and  $k$  are used in the calculation of  $s = ex + k$ . The tamper resistant device stores  $x, 1/x, k, 1/k$  somewhere before the calculation starts. After computing  $s' = ex' + k'$ , the device checks to see if  $e = k(s'(1/k) - 1)(1/x)$ . The value  $s'$  is considered correct only if the equality holds.

## 4 Concluding Remarks

The attack to public-key cryptographic schemes on tamper resistant devices presented in this paper makes use of transient faults. Our attacking model is independent of the implementation of a specific cryptosystem and seems to be applicable to breaking large classes of public-key cryptosystems. In particular, we showed how to break the RSA, the ElGamal signature scheme, the Schnorr signature scheme, and the DSA.

This attack highlighted that hardware faults can cause fatal leakage of the private/secret key values and may eventually lead to breaking of a cryptosystem. Therefore, it is important to take fault tolerance into serious consideration in the design of cryptosystems and to strike a balance between low overhead and high robustness. As a first step, we have proposed some methods to counter our attack. It should be noted that there are many other ways of breaking tamper resistant devices in addition to the ones outlined in this paper. In general, design of fault tolerant tamper resistant devices is a very challenging problem. Readers interested in getting more information in this area are referred to the excellent paper by Anderson and Kuhn [1].

## 5 Acknowledgments

The work reported in this paper was motivated by the Bellcore Press Release [4] and further motivated by Biham and Shamir's research announcement [5].

The fundamental concept of breaking cryptosystems in the presence of hardware faults released in [4] was the main driving force behind our research. The authors would also like to thank Dr. Arjen Lenstra for his valuable comments on an earlier version of our manuscript.

## References

1. R. Anderson and M. Kuhn, "Tamper Resistance - A Cautionary Note", to appear in the Proceedings of the 2nd Workshop on Electronic Commerce, Oakland, CA., Nov. 18-20, 1996.
2. F. Bao, R. Deng, Y. Han, A. Jeng, D. Narasimhalu, and T. Ngair, "Another New Attack to RSA on Tamperproof Devices", 23rd October. 1996, <http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/news-items/961022.sgtamper.html>; "A Method to Counter Another New Attack to RSA on Tamperproof Devices", 24th October. 1996, <http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/news-items/961024.sgtampercounter.html>.
3. F. Bao, R. Deng, Y. Han, A. Jeng, D. Narasimhalu, and T. Ngair, "New Attacks to Public Key Cryptosystems on Tamperproof Devices", 29th October. 1996, <http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/news-items/>.
4. Bellcore Press Release, "New Threat Model Breaks Crypto Codes", Sept. 1996, <http://www.bellcore.com/PRESS/ADVSR96/facts.html>.
5. E. Biham and A. Shamir, "Research Announcement: A New Cryptanalytic Attack on DES", 18th October 1996, <http://jya.com/dfa.htm>.
6. D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the Importance of Checking Computations", Submitted to Eurocrypt 96.
7. T. ElGamal, "A Public-Key Cryptosystems and a Signature Scheme Based on Discrete Logarithms", IEEE Trans. Information Theory, Vol. IT-31, No. 4, 1985, pp. 469-472.
8. M. Joye and J.-J. Quisquater, "Attacks on systems using Chinese remaindering", Technical Report CG-1996/9 of UCL, <http://www.dice.ucl.ac.be/crypto/>.
9. A. K. Lenstra, "Memo on RSA Signature Generation in the Presence of Faults", Manuscript, Sept. 28, 1996. Available from Author at [arjen.lenstra@citicorp.com](mailto:arjen.lenstra@citicorp.com).
10. R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, vol. 21, No. 2, Feb. 1978, pp. 120-126.
11. C. Schnorr, "Efficient Signature Generation by Smart Cards", J. Cryptology, Vol. 4, 1991, pp. 161-174.